SMORE: Similarity-based Hyperdimensional Domain Adaptation for Multi-Sensor Time Series Classification

Junyao Wang, Mohammad Abdullah Al Faruque
{junyaow4,alfaruqu}@uci.edu
Department of Computer Science, University of California, Irvine, USA

ABSTRACT

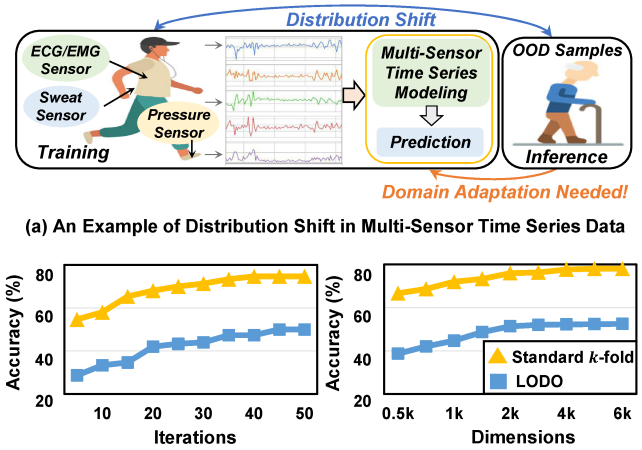
Many real-world applications of the Internet of Things (IoT) employ machine learning (ML) algorithms to analyze time series information collected by interconnected sensors. However, *distribution shift* arises when a model is deployed on a data distribution different from the training data and can substantially degrade model performance. Additionally, increasingly sophisticated deep neural networks (DNNs) are required to capture intricate spatial and temporal dependencies in multi-sensor time series data, often exceeding the capabilities of edge devices. We propose SMORE, a novel resource-efficient domain adaptation (DA) algorithm for multi-sensor time series classification, leveraging the efficient and parallel operations of hyperdimensional computing. SMORE dynamically customizes test-time models with explicit consideration of the domain context of each sample to mitigate the negative impacts of domain shifts. Our evaluation shows that SMORE achieves on average 1.98% higher accuracy than state-of-the-art (SOTA) DNN-based DA algorithms with 18.81× faster training and 4.63× faster inference.

ACM Reference Format:
Junyao Wang, Mohammad Abdullah Al Faruque. 2024. SMORE: Similarity-based Hyperdimensional Domain Adaptation for Multi-Sensor Time Series Classification. In *61st ACM/IEEE Design Automation Conference (DAC '24), June 23–27, 2024, San Francisco, CA, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3649329.3658477

1 INTRODUCTION

Many real-world applications utilize heterogeneously connected sensors to collect information over the course of time, constituting multi-sensor time series data [1]. Machine learning (ML) algorithms, including deep neural networks (DNNs), are often employed to analyze the collected data and perform various learning tasks. However, the excellent performance of these ML algorithms heavily relies on the key assumption that the training and inference data come from the same distribution, while this assumption can be easily violated as out-of-distribution (OOD) scenarios are inevitable in real-world applications [2, 3]. For instance, as shown in Figure 1(a), a human activity recognition model can systematically fail when tested on individuals from different age groups or diverse demographics.

Various novel *domain adaptation* techniques have been proposed for deep learning (DL) [4–7]. However, due to their limited capacity for memorization, DL approaches often fail to perform well on multi-sensor time series data with intricate spatial and temporal dependencies [1, 8]. Although recurrent neural networks (RNNs), including long short-term memory (LSTM), have recently been proposed to address this issue, these models are notably complicated and inefficient to train. Specifically, their complex architectures require iteratively refining millions of parameters over multiple time periods in powerful computing environments [9, 10]. Considering the resource constraints of embedded devices, the massive amount of information nowadays, and the potential instabilities of IoT systems, a lightweight and efficient domain-adaptive learning approach for multi-sensor time series data is critically needed.



(a) An Example of Distribution Shift in Multi-Sensor Time Series Data

(b) Comparing LODO CV and Standard k-fold CV of SOTA HDC

Figure 1: Motivation of Our Proposed SMORE

Hyperdimensional computing (HDC) has been introduced as a promising paradigm for edge ML for its high computational efficiency and ultra-robustness against noises [8, 11, 12]. In particular, HDC incorporates learning capability along with typical memory functions of storing/loading information, bringing unique advantages in tackling noisy time series [8]. Unfortunately, existing HDCs are vulnerable to DS. For instance, as shown in Figure 1(b), on the popular multi-sensor time series dataset USC-HAD [13], SOTA HDCs converge at notably lower accuracy in leave-one-domain-out (LODO) cross-validation (CV) than in standard k-fold CV regardless of training iterations and model complexity. LODO CV develops a model with all the available data except for one domain left out for inference, while standard k-fold CV randomly divides data into k subsets with $k-1$ subsets for training and the remaining one for evaluation. Such performance degradation indicates a very limited domain adaptation (DA) capability of existing HDCs. However, standard k-fold CV does not reflect real-world DS issues since the random sampling process introduces *data leakage*, enabling the

DAC '24, June 23–27, 2024, San Francisco, CA

ACM ISBN 979-8-4007-0601-1/24/06.
https://doi.org/10.1145/3649329.3658477

training phase to include information from all domains , thus inflating model performance. To address this problem, we propose SMORE, a novel HDC-based DA algorithm for multi-sensor time series classification. Our main contributions are listed below:

- To the best of our knowledge, SMORE is the first HDC-based DA algorithm. By explicitly considering the domain context of each sample during inference, SMORE provides on average 1.98% higher accuracy than SOTA DL-based DA algorithms on a wide range of multi-sensor time series classification tasks.
- Leveraging the efficient and parallel high-dimensional operations, SMORE provides on average 18.81× faster training and 4.63× faster inference than SOTA DL-based DA algorithms, providing a more efficient solution to tackle the DS challenge.
- We evaluate SMORE across multiple resource-constrained hardware devices including Raspberry Pi and NVIDIA Jetson Nano. SMORE demonstrates considerably lower inference latency and energy consumption than SOTA DL-based DA algorithms.

2 RELATED WORKS

2.1 Domain Adaptation

Distribution shift (DS) arises when a model is deployed on a data distribution different from what it was trained on, posing serious robustness challenges for real-world ML applications [4, 5]. Methodologies addressing DS can be primarily categorized as *domain generalizations* (DG) and *domain adaptations* (DA). DG seeks to construct models by identifying domain-invariant features shared across multiple source domains [14]. However, it can be challenging to extract common features when there exist multiple source domains with distinct characteristics [6, 8, 15]. Thus, existing DG approaches often fail to provide comparable high-quality results as DA [7, 16]. In contrast, DA generally utilizes unlabeled data in target domains to quickly adapt models trained in different source domains [4]. Unfortunately, most existing DA techniques rely on multiple convolutional and fully-connected layers to train domain discriminators, requiring intensive computations and iterative refinement.

2.2 Hyperdimensional Computing

Several recent works have utilized HDC as a lightweight learning paradigm for time series classification [17–19], achieving comparable accuracy to SOTA DNNs with lower computational costs. However, existing HDCs do not consider the DS challenge, which can be a detrimental drawback for real-world ML applications. Hyperdimensional Feature Fusion [3] identifies OOD samples by fusing outputs of several layers of a DNN model to a common hyperdimensional space, while its backbone remains relying on resource-intensive multi-layer DNNs, and a systematic way to tackle OOD samples has yet to be proposed. DOMINO [8], a novel HDC-based DG method, constantly discards and regenerates biased dimensions representing domain-variant information; nevertheless, it requires significantly more training time to provide reasonable accuracy. Our proposed SMORE leverages efficient operations of HDC to customize test-time models for OOD samples, providing accurate predictions without causing substantial computational overhead for both training and inference.

3 METHODOLOGY

3.1 HDC Prelimnaries

Inspired by high-dimensional information representation of human brains, HDC maps inputs onto *hyperdimensional* space as *hypervectors*, each containing thousands of elements. A unique property of the hyperdimensional space is the existence of large amounts of nearly orthogonal hypervectors, enabling highly parallel and efficient operations. **Similarity (δ):** calculation of the distance between two hypervectors. A common measure is cosine similarity. **Bundling (+):** element-wise addition of hypervectors, e.g., $\mathcal{H}_{bundle} = \mathcal{H}_1 + \mathcal{H}_2$, generating a hypervector with the same dimension as inputs. Bundling provides an efficient way to check the existence of a hypervector in a bundled set. In the previous example, $\delta(\mathcal{H}_{bundle}, \mathcal{H}_1) \gg 0$ while $\delta(\mathcal{H}_{bundle}, \mathcal{H}_3) \approx 0\, (\mathcal{H}_3 \neq \mathcal{H}_1, \mathcal{H}_2)$. Bundling models how human brains *memorize* inputs. **Binding ($*$):** element-wise multiplication of two hypervectors to create another near-orthogonal hypervector, i.e. $\mathcal{H}_{bind} = \mathcal{H}_1 * \mathcal{H}_2$ where $\delta(\mathcal{H}_{bind}, \mathcal{H}_1) \approx 0$ and $\delta(\mathcal{H}_{bind}, \mathcal{H}_2) \approx 0$. Due to reversibility, i.e., $\mathcal{H}_{bind} * \mathcal{H}_1 = \mathcal{H}_2$, information from both hypervectors can be preserved. Binding models how human brains *connect* inputs. **Permutation** (ρ): a single circular shift of a hypervector by moving the value of the final dimension to the first position and shifting all other values to their next positions. The permuted hypervector is nearly orthogonal to its original hypervector, i.e., $\delta(\rho\mathcal{H}, \mathcal{H}) \approx 0$. Permutation models how human brains handle *sequential* inputs.

3.2 Problem Formulation

We assume that there are $\mathcal{K}$ ($\mathcal{K} > 1$) source domains, i.e., $\mathcal{D}_S = \{\mathcal{D}_S^1, \mathcal{D}_S^2, \ldots, \mathcal{D}_S^{\mathcal{K}}\}$, in the input space $\mathcal{I}$, and we denote the output space as $\mathcal{Y}$. $\mathcal{I}$ consists of time-series data from m ($m \geq 1$) interconnected sensors, i.e., $\mathcal{I} = \{\mathcal{I}_1, \mathcal{I}_2, \ldots, \mathcal{I}_m\}$. Our objective is to utilize training samples in $\mathcal{I}$ and their corresponding labels in $\mathcal{Y}$ to train a classification model $f : \mathcal{I} \rightarrow \mathcal{Y}$ to capture latent features so that we can make accurate predictions when given samples from an unseen target domain $\mathcal{D}_\mathcal{T}$. The key challenge is that the joint distribution between source domains and target domains can be different, i.e, $\mathcal{P}^{\mathcal{S}}_{(\mathcal{I},\mathcal{Y})} \neq \mathcal{P}^{\mathcal{T}}_{(\mathcal{I},\mathcal{Y})}$, and thus the model f can potentially fail to adapt models trained on $\mathcal{D}_\mathcal{S}$ to samples from $\mathcal{D}_\mathcal{T}$.

Our proposed SMORE, as shown in Figure 2, starts with mapping training samples from the input space $\mathcal{I}$ to a hyperdimensional space $\mathcal{X}$ with an encoder Ω (Ⓐ), i.e., $\mathcal{X} = \Omega(\mathcal{I})$, that preserves the spatial and temporal dependencies in $\mathcal{I}$. We then separate the encoded data into $\mathcal{K}$ subsets $(\mathcal{X}_1, \mathcal{Y}_1), (\mathcal{X}_2, \mathcal{Y}_2), \ldots, (\mathcal{X}_\mathcal{K}, \mathcal{Y}_\mathcal{K})$ (Ⓑ) based on their domains. In the training phase, we train $\mathcal{K}$ *domain-specific* models $\mathcal{M} = \{\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_\mathcal{K}\}$ such that $\mathcal{M}_k : \mathcal{X}_k \rightarrow \mathcal{Y}_k$ ($1 \leq k \leq \mathcal{K}$) (Ⓒ), and concurrently develop $\mathcal{K}$ expressive *domain descriptors* $\mathcal{U} = \{\mathcal{U}_1, \mathcal{U}_2, \ldots, \mathcal{U}_\mathcal{K}\}$ encoding the pattern of each domain (Ⓓ). When given a inference sample $\mathcal{I}_\mathcal{T}$ from the target domain $\mathcal{D}_\mathcal{T}$, we map $\mathcal{I}_\mathcal{T}$ to hyperdimensional space with the same encoder Ω used for training, and identify OOD samples (Ⓔ) with a binary classifier Φ utilizing the domain descriptors $\mathcal{U}$, i.e., $\Phi(\Omega(\mathcal{I}_\mathcal{T}), \mathcal{U})$. Finally, we construct a *test-time model* $\mathcal{M}_\mathcal{T}$ based on domain-specific models $\mathcal{M}$ (Ⓕ) and whether the sample is OOD to make inferences (Ⓖ) with explicit consideration of the domain context of $\mathcal{I}_\mathcal{T}$, i.e., $\hat{\mathcal{Y}}_\mathcal{I} = \mathcal{M}_\mathcal{T}(\Phi(\Omega(\mathcal{I}_\mathcal{T}), \mathcal{U}), \mathcal{M})$.

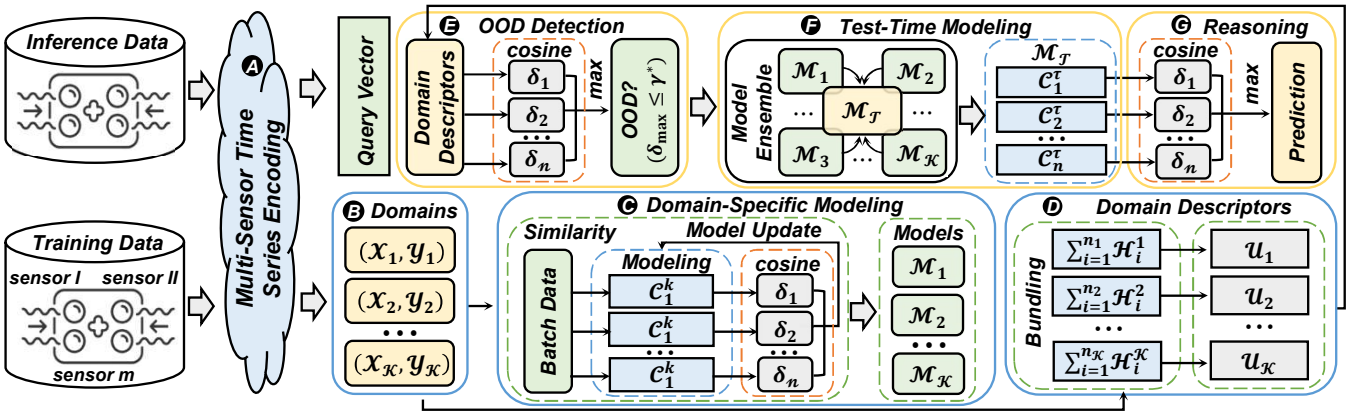


Figure 2: The Workflow of Our Proposed SMORE

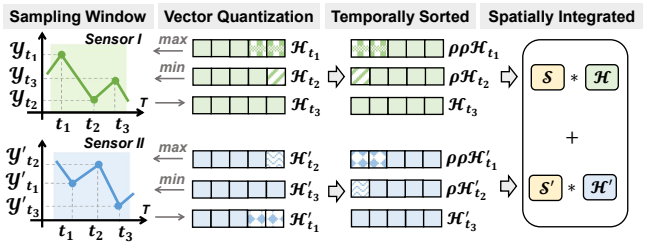


Figure 3: HDC Encoding for Multi-Sensor Time Series Data

3.3 Multi-Sensor Time Series Data Encoding

We employ the encoding techniques in Figure 3 to capture and preserve spatial and temporal dependencies in multi-sensor time series data. We sample time series data in n-gram windows; in each sample window, the signal values (y-axis) store the information and the time (x-axis) represents the temporal sequence. We first assign random hypervectors $\mathcal{H}_{max}$ and $\mathcal{H}_{min}$ to represent the maximum and minimum signal values. We then perform vector quantization to values between the maximum and minimum values to generate vectors with a spectrum of similarity to $\mathcal{H}_{max}$ and $\mathcal{H}_{min}$. For instance, in Figure 3, Sensor I has the maximum value at t_1 and the minimum value at t_2, and thus we assign randomly generated hypervectors $\mathcal{H}_{t_1}$ and $\mathcal{H}_{t_2}$ to y_{t_1} and y_{t_2}, respectively. Similarly, we assign randomly generated hypervectors $\mathcal{H}'_{t_2}$ and $\mathcal{H}'_{t_3}$ to y'_{t_2} and y'_{t_3} in Sensor II. We then assign hypervectors to y_{t_3} in Sensor I and value at y'_{t_1} in Sensor II with vector quantization; mathematically,

$$\mathcal{H}_{t_3} = \mathcal{H}_{t_2} + \frac{y_{t_3} - y_{t_2}}{y_{t_1} - y_{t_2}} \cdot (\mathcal{H}_{t_1} - \mathcal{H}_{t_2})$$
$$\mathcal{H}'_{t_1} = \mathcal{H}'_{t_3} + \frac{y'_{t_1} - y'_{t_3}}{y'_{t_2} - y'_{t_3}} \cdot (\mathcal{H}'_{t_2} - \mathcal{H}'_{t_3}).$$

We represent the temporal sequence of data with the permutation operation in section 3.1. For Sensor I and Sensor II in Figure 3, we perform rotation shift (ρ) twice to $\mathcal{H}_{t_1}$ and $\mathcal{H}'_{t_1}$, once to $\mathcal{H}_{t_2}$ and $\mathcal{H}'_{t_2}$, and keep $\mathcal{H}_{t_3}$ and $\mathcal{H}'_{t_3}$ the same. We bind data samples in one sampling widow by calculating $\mathcal{H} = \rho\rho\mathcal{H}_{t_1} * \rho\mathcal{H}_{t_2} * \mathcal{H}_{t_3}$ and $\mathcal{H}' = \rho\rho\mathcal{H}'_{t_1} * \rho\mathcal{H}'_{t_2} * \mathcal{H}'_{t_3}$. Finally, to spatially integrate data from multiple sensors, we generate a random signature hypervector for each sensor and bind information as $\sum_{i=1}^{m}[\mathcal{G}_i * \mathcal{H}_i]$, where $\mathcal{G}_i$ denotes the signature hypervector for sensor i, $\mathcal{H}_i$ is the hypervector containing overall information from sensor i, and m denotes the total number of sensors. For our example in Figure 3, we combine information from Sensor I and Sensor II by randomly generating signature hypervectors $\mathcal{G}$ and $\mathcal{G}'$ and calculating $(\mathcal{G} * \mathcal{H}) + (\mathcal{G}' * \mathcal{H}')$.

3.4 Domain-Specific Modeling

As shown in Figure 2, after mapping data to hyperdimensional space (A), SMORE separates training samples into $\mathcal{K}$ subsets based on their domains (B), where $\mathcal{K}$ represents the total number of domains.. We then employ a highly efficient HDC algorithm to calculate a *domain-specific* model for every domain (C), i.e., $\mathcal{M} = \{\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_\mathcal{K}\}$, identifying common patterns during training while eliminating model saturations. We bundle data points by scaling a proper weight to each of them depending on how much new information is added to class hypervectors. In particular, each domain-specific model $\mathcal{M}_k (1 \le k \le \mathcal{K})$ consists of n class hypervectors $C_1^k, C_2^k, \ldots, C_n^k$ (n = the number of classes), each of which encodes the pattern of a class. A new sample $\mathcal{H}$ in domain $\mathcal{D}_\mathcal{S}^k$ updates model $\mathcal{M}_k$ based on its cosine similarities, denoted as $\delta(\mathcal{H}, \cdot)$, with all the class hypervectors in $\mathcal{M}_k$, i.e.,

$$\delta(\mathcal{H}, C_t^k) = \frac{\mathcal{H} \cdot C_t^k}{\|\mathcal{H}\| \cdot \|C_t^k\|} = \frac{\mathcal{H}}{\|\mathcal{H}\|} \cdot \frac{C_t^k}{\|C_t^k\|} \propto \mathcal{H} \cdot Norm(C_t^k) \quad (1)$$

where $1 \le t \le n$. If $\mathcal{H}$ has the highest cosine similarity with the class hypervector C_i^k while its true label C_j^k $(1 \le i, j \le n)$, the domain-specific model $\mathcal{M}_k$ updates as

$$C_j^k \leftarrow C_j^k + \eta \cdot \left(1 - \delta(\mathcal{H}, C_j^k)\right) \times \mathcal{H}$$
$$C_i^k \leftarrow C_i^k - \eta \cdot \left(1 - \delta(\mathcal{H}, C_i^k)\right) \times \mathcal{H}, \quad (2)$$

where η denotes a learning rate. A large $\delta(\mathcal{H}, \cdot)$ indicates the input data point is marginally mismatched or already exists in the model, and the model is updated by adding a very small portion of the encoded vector $(1 - \delta(\mathcal{H}, \cdot) \approx 0)$. In contrast, a small $\delta(\mathcal{H}, \cdot)$ indicates a noticeably new pattern and updates the model with a large factor $(1 - \delta(\mathcal{H}, \cdot) \approx 1)$. Our learning algorithm provides a higher chance for non-common patterns to be properly included in the model.

3.5 Out-of-Distribution Detection

3.5.1 Domain Descriptors. In parallel to domain-specific modeling, as shown in Figure 2, we concurrently construct *domain descriptors* (D) $\mathcal{U} = \{\mathcal{U}_1, \mathcal{U}_2, \ldots, \mathcal{U}_\mathcal{K}\}$ to encode the distinct pattern of each domain. Specifically, for each domain $\mathcal{D}_\mathcal{S}^k$ $(1 \le k \le \mathcal{K})$, we utilize the bundling operation to combine the hypervector within the domain, i.e., $\{\mathcal{H}_1^k, \mathcal{H}_2^k, \ldots, \mathcal{H}_{n_k}^k\}$, and construct an expressive descriptor $\mathcal{U}_k$. Mathematically, $\mathcal{U}_k = \sum_i^{n_k} \mathcal{H}_i^k$. Given the property of the bundling operation (explained in section 3.1), $\mathcal{U}_k$ is cosine-similar to all the samples $\mathcal{H}_i^k$ $(1 \le i \le n_k)$ within domain $\mathcal{D}_\mathcal{S}^k$ as they contribute to the bundling process, and dissimilar to all the samples that are not part of the bundle, i.e., not in domain $\mathcal{D}_\mathcal{S}^k$.

3.5.2 Out-of-Distribution Detection. As shown in Figure 2, a key component of our inference phase is the detection of OOD samples (E). We start with mapping the testing sample to hyperdimensional space with the same encoding technique as the training phase to obtain a query vector $\mathcal{Q}$ (A). Then, as detailed in Algorithm 1, we calculate the cosine similarity score of the query vector $\mathcal{Q}$ to each domain descriptor $\mathcal{U}_1, \mathcal{U}_2, \ldots, \mathcal{U}_\mathcal{K}$ (line 1). A testing sample is identified as OOD if its pattern is substantially different from all the source domains. Therefore, when the cosine similarity between $\mathcal{Q}$ and its most similar domain, i.e., $\max\{\delta(\mathcal{Q}, \mathcal{U}_1), \delta(\mathcal{Q}, \mathcal{U}_2), \ldots, \delta(\mathcal{Q}, \mathcal{U}_\mathcal{K})\}$, is smaller than a threshold δ^*, we consider $\mathcal{Q}$ as an OOD sample (line 2). Here δ^* is a tunable parameter and we analyze the impact of δ^* in section 4.2.1. We then dynamically construct test-time models (F) based on whether the sample is OOD (section 3.6).

3.6 Adaptive Test-Time Modeling

3.6.1 Inference for OOD Samples. For each testing sample identified as OOD, we dynamically adapt domain-specific models $\mathcal{M} = \{\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_k\}$ to customize a test-time model $\mathcal{M}_\mathcal{T}$ that best fits its domain context and thereby provides an accurate prediction. As detailed in Algorithm 1, for an OOD sample $\mathcal{Q}$, we ensemble each domain-specific model based on how similar $\mathcal{Q}$ is to the domain (line 3). Specifically, let $\delta(\mathcal{Q}, \mathcal{U}_1)$, $\delta(\mathcal{Q}, \mathcal{U}_2)$, $\ldots$, $\delta(\mathcal{Q}, \mathcal{U}_\mathcal{K})$ denote the cosine similarity between $\mathcal{Q}$ and each domain descriptor $\mathcal{U}_1, \mathcal{U}_2, \ldots, \mathcal{U}_\mathcal{K}$, we construct the test-time model $\mathcal{M}_\mathcal{T}$ for $\mathcal{Q}$ as

$$\mathcal{M}_\mathcal{T} = \delta(\mathcal{Q}, \mathcal{U}_1) \cdot \mathcal{M}_1 + \delta(\mathcal{Q}, \mathcal{U}_2) \cdot \mathcal{M}_2 + \ldots + \delta(\mathcal{Q}, \mathcal{U}_\mathcal{K}) \cdot \mathcal{M}_\mathcal{K}. \quad (3)$$

Specifically, $\mathcal{M}_\mathcal{T}$ is of the same shape as $\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_k$; it consists of n class hypervectors (n = number of classes), denoted as $C_1^\mathcal{T}, C_2^\mathcal{T}, \ldots, C_n^\mathcal{T}$, formulated with explicit consideration of the domain context of $\mathcal{Q}$. We then compute the cosine similarity between $\mathcal{Q}$ and each of these class hypervectors, and assign $\mathcal{Q}$ to the class to which it achieves the highest similarity score (line 7).

3.6.2 Inference for In-Distribution Samples. We predict the label of an in-distribution testing sample, i.e., a non-OOD sample, leveraging domain-specific models of the domains to which the sample is highly similar. As demonstrated in Algorithm 1, for all the domains $\mathcal{D}_\mathcal{S}^i$ $(1 \le i \le \mathcal{K})$ where the encoded query vector $\mathcal{Q}$ achieves a similarity score higher than δ^*, we ensemble their corresponding domain-specific models incorporating a weight of their cosine similarity score with $\mathcal{Q}$ to formulate the test-time model $\mathcal{M}_\mathcal{T}$ (line 5 - 6). Similar to section 3.6.1, $\mathcal{M}_\mathcal{T}$ consists of n class hypervectors, and $\mathcal{Q}$ is assigned to the class where it obtained the highest similarity score

Algorithm 1 Domain Adaptive HDC Inference

Input: An encoded testing samples $\mathcal{Q}$, a domain classifier with $\mathcal{K}$ class hypervectors $\mathcal{U}_1, \mathcal{U}_2, \ldots, \mathcal{U}_\mathcal{K}$, a threshold for OOD detection δ^*, $\mathcal{K}$ domain-specific models $\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_\mathcal{K}$
Output: A predicted label $\mathcal{P}$ for $\mathcal{Q}$.

1: $\delta_{max} = \max\{\delta(\mathcal{Q}, \mathcal{U}_1), \delta(\mathcal{Q}, \mathcal{U}_2) \ldots, \delta(\mathcal{Q}, \mathcal{U}_\mathcal{K})\}$ ▷ OOD detection
2: **if** $\delta_{max} < \delta^*$ **then** ▷ $\mathcal{Q}$ is considered OOD
3: $\quad \mathcal{M}_\mathcal{T} \leftarrow \sum_{i=1}^{\mathcal{K}} \delta(\mathcal{Q}, \mathcal{U}_i) \cdot \mathcal{M}_i$ ▷ model ensembling
4: **else** ▷ $\mathcal{Q}$ is not OOD
5: $\quad$ **for all** $\delta(\mathcal{Q}, \mathcal{U}_i) \ge \delta^*$ **do** ▷ $1 \le i \le k$
6: $\quad\quad \mathcal{M}_\mathcal{T} \leftarrow \sum_{i=1}^{\mathcal{K}} \delta(\mathcal{Q}, \mathcal{U}_i) \cdot \mathcal{M}_i$ ▷ partial model ensembling
7: $\mathcal{P} \leftarrow \arg\max_{C_i^\mathcal{T}} \{\delta(\mathcal{Q}, C_1^\mathcal{T}), \delta(\mathcal{Q}, C_2^\mathcal{T}) \ldots, \delta(\mathcal{Q}, C_n^\mathcal{T})\}$
8: **return** $\mathcal{P}$

Table 1: Detailed Breakdowns of Datasets
($\mathcal{N}$: number of data samples)

DSADS [20]		USC-HAD [13]		PAMAP2 [21]	
Domains	$\mathcal{N}$	Domains	$\mathcal{N}$	Domains	$\mathcal{N}$
Domain 1	2,280	Domain 1	8,945	Domain 1	5,636
Domain 2	2,280	Domain 2	8,754	Domain 2	5,591
Domain 3	2,280	Domain 3	8,534	Domain 3	5,806
Domain 4	2,280	Domain 4	8,867	Domain 4	5,660
		Domain 5	8,274		
Total	9,120	Total	43,374	Total	22,693

(line 7). Note that, unlike the inference for OOD samples where we ensemble all the domain-specific models to enhance performance, the test-time model $\mathcal{M}_\mathcal{T}$ for an in-distribution sample does not consider domain-specific models of the domains where $\mathcal{Q}$ show a minor similarity score lower than δ^*. In particular, if a testing sample $\mathcal{Q}$ does not show high similarity to any of the source domains, we include information from all the domains to construct a sufficiently comprehensive test-time model to mitigate the negative impacts of distribution shift. In contrast, when $\mathcal{Q}$ exhibits considerable similarity to certain domains, adding information from other domains is comparable to introducing noises and can potentially mislead the classification and degrade model performance.

4 EXPERIMENTAL EVALUATIONS

4.1 Experimental Setup

We evaluate SMORE on widely-used multi-sensor time series datasets DSADS [20], USC-HAD [13], PAMAP2 [21]. Domains are defined by subject grouping chosen based on subject ID from low to high. The data size of each domain in each dataset is demonstrated in TABLE 1. We compare SMORE with (i) two SOTA CNN-based DA algorithms: TENT [4] and multisource domain adversarial networks (MDANs) [5], and (ii) two HDC algorithms: the SOTA HDC not considering distribution shifts [22] (BaselineHD) and DOMINO [8], a recently proposed HDC-based domain generalization framework. The CNN-based DA algorithms are trained with TensorFlow, and we apply the common practice of grid search to identify the best hyper-parameters for each model. Since DOMINO involves dimension regeneration in every training iteration, for fairness, we initiate it with dimension $d^* = 1k$ and make its total dimensionalities, i.e., the sum of its initial dimension and all regenerated dimensions throughout retraining, the same as SMORE and BaselineHD.

4.1.1 Platforms. To evaluate the performance of SMORE on both high-performance computing environments and resource-limited edge devices, we include results from the following platforms:

- **Server CPU**: Intel Xeon Silver 4310 CPU (12-core, 24-thread, 2.10 GHz), 96 GB DDR4 memory, Ubuntu 20.04, Python 3.8.10, PyTorch 1.12.1, TDP 120 W.
- **Embedded CPU**: Raspberry Pi 3 Model 3+ (quad-core ARM A53 @1.4GHz), 1 GB LPDDR2 memory, Debian 11, Python 3.9.2, PyTorch 1.13.1, TDP 5 W.
- **Embedded GPU**: Jetson Nano (quad-core ARM A57 @1.43 GHz, 128-core Maxwell GPU), 4 GB LPDDR4 memory, Python 3.8.10, PyTorch 1.13.0, CUDA 10.2, TDP 10 W.

4.1.2 Data Preprocessing. We describe the data processing for each dataset primarily on data segmentation and domain labeling.

- **DSADS**[20]: The Daily and Sports Activities Dataset (DSADS) includes 19 activities performed by 8 subjects. Each data segment is a non-overlapping five-second window sampled at 25Hz. Four domains are formed with two subjects each.
- **USC-HAD** [13]: The USC human activity dataset (USC-HAD) includes 12 activities performed by 14 subjects. Each data segment is a 1.26-second window sampled at 100Hz with 50% overlap. Five domains are formed with three subjects each.
- **PAMAP2** [21]: The Physical Activity Monitoring (PAMAP2) dataset includes 18 activities performed by 9 subjects. Each data segment is a 1.27-second window sampled at 100Hz with 50% overlap. Four domains, excluding subject nine, are formed with two subjects each.

4.2 Accuracy

The accuracy of the LODO classification is demonstrated in Figure 4. The accuracy of Domain k indicates that the model is trained with data from all the other domains and evaluated with data from Domain k. This accuracy score serves as an indicator of the model's domain adaptation capability when confronted with unseen data from unseen distributions. SMORE achieves comparable performance to TENT and on average 1.98% higher accuracy than MDANs. Additionally, SMORE provides 20.25% higher accuracy than BaselineHD, demonstrating that SMORE successfully adapts trained models to fit samples from diverse domains. Additionally, SMORE provides on average 4.56% higher accuracy than DOMINO, indicating DA can more effectively address the domain distribution (DS) challenge in noisy multi-sensor time series data compared to DG.

4.2.1 Impact of Hyperparameter δ^*. The impact of δ^* on classification results is demonstrated in Figure 5, where we evaluate SMORE on the dataset USC-HAD as an example. SMORE achieved its best performance when δ^* is around 0.65. Small values of δ^* are more likely to detect in-distribution samples as OOD so that the test-time models would include noises from the domains where the sample only has a minimal similarity score, thereby causing notable performance degradation. On the other hand, large values of δ^* are more likely to consider OOD samples in-distribution. Consequently, the test-time model only includes domain-specific models of the domains that exhibit limited similarity to the given sample; therefore, the ensembled test-time model is unlikely to be sufficiently comprehensive to provide accurate predictions.

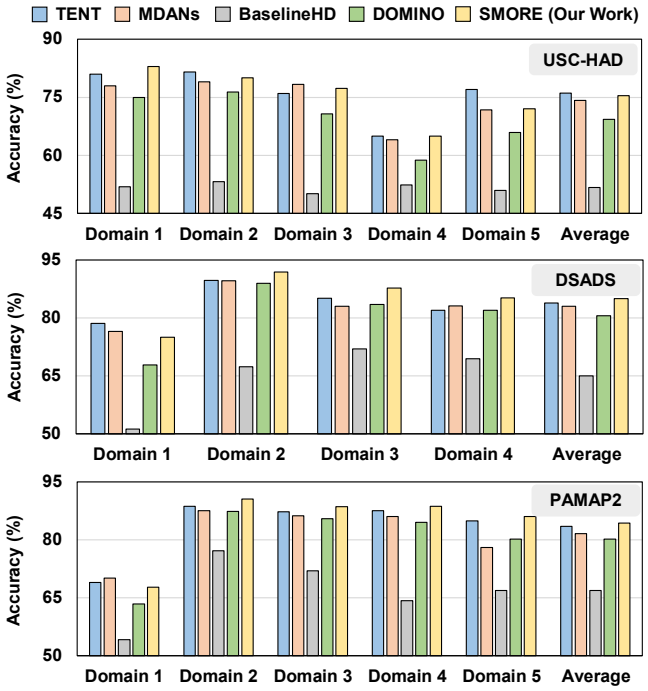


Figure 4: Comparing LODO Accuracy of SMORE and CNN-based Domain Adaptation Algorithms

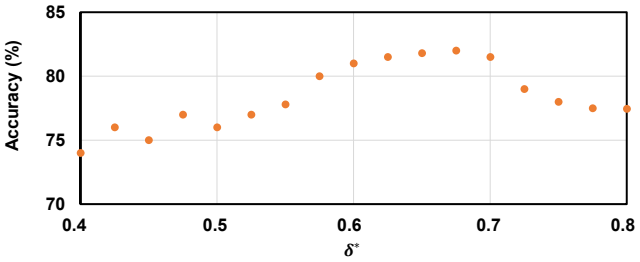


Figure 5: Imapct of δ^* on Model Performance

4.3 Efficiency

4.3.1 Efficiency on Server CPU. For each dataset, each domain consists of roughly similar amounts of data as detailed in TABLE 1; thus, we show the average runtime of training and inference for all the domains. As demonstrated in Figure 6(a), SMORE exhibits on average 11.64× faster training than TENT and 18.81× faster training than MDANs. Additionally, SMORE delivers 4.07× faster inference than TENT and 4.63× faster inference than MDANs. Such notably higher learning efficiency is thanks to the highly parallel matrix operations on hyperdimensional space. Additionally, SMORE provides on average 5.84× faster training compared to DOMINO. In particular, DOMINO achieves domain generalization by iteratively identifying and regenerating domain-variant dimensions and thus requires notably more retraining iterations to provide reasonable performance. On the other hand, during each retraining iteration, DOMINO only keeps dimensions playing the most positive role in the classification task, and thus it eventually arrives at a model with compressed dimensionality and provides a slightly higher inference

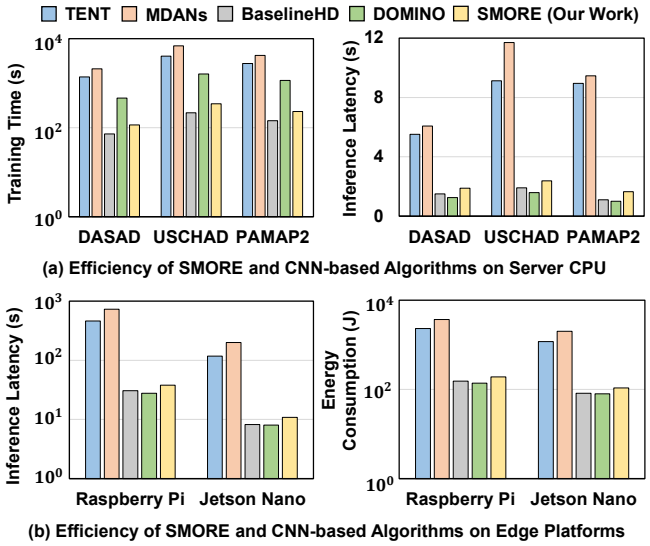


Figure 6: Efficiency of SMORE and CNN-based DA Algorithms

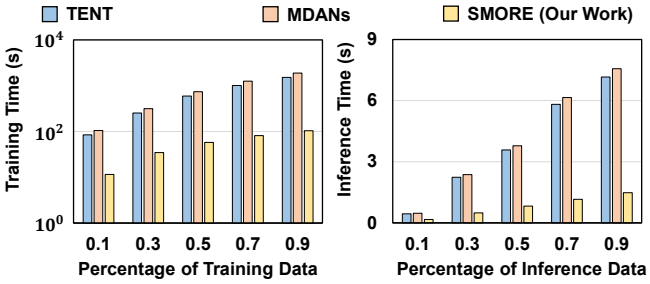


Figure 7: Comparing Scalability Using Different Size of Data

efficiency. Furthermore, compared to BaselineHD, SMORE provides significantly higher accuracy (demonstrated in Fig. 4) without substantially increasing both the training and inference runtimes.

4.3.2 Efficiency on Embedded Platforms. To further understand the performance of SMORE on resource-constrained computing platforms, we evaluate the efficiency of SMORE, TENT, MDANs, and BaselineHD using a Raspberry Pi 3 Model B+ board and an NVIDIA Jetson Nano board. Both platforms have very limited memory and CPU cores (and GPU cores for Jetson Nano). Figure 6(b) shows the average inference latency for each algorithm processing each domain in the PAMAP2 dataset. On Raspberry Pi, SMORE provides on average14.82× speedups compared to TENT and 19.29× speedups compared to MDANs in inference. On Jetson Nano, SMORE delivers 13.22× faster inference than TENT and 17.59× faster inference than MDANs. Additionally, SMORE exhibits significantly less energy consumption, providing a more resource-efficient domain adaptation solution for the distribution shift challenge on edge devices.

4.4 Scalability

We evaluate the scalability of SMORE and SOTA CNN-based DA algorithms using various training data sizes (percentages of the full dataset) of the PAMAP2 dataset. As demonstrated in Figure 7, with increasing the training data size, SMORE maintains high efficiency in both training and inference with a sub-linear growth in execution time. In contrast, the training and inference time of CNN-based DA algorithms increases considerably faster than SMORE. This positions SMORE as an efficient and scalable DA solution for both high-performance and resource-constrained computing platforms.

5 CONCLUSIONS

We propose SMORE, a novel HDC-based domain adaptive algorithm that dynamically customizes test-time models with explicit consideration of the domain context of each sample and thereby provides accurate predictions. SMORE outperforms SOTA CNN-based DA algorithms in both accuracy and training and inference efficiency, making it an ideal solution to address the DS challenge.

6 ACKNOWLEDGEMENT

This work was partially supported by the National Science Foundation (NSF) under award CCF-2140154.

REFERENCES

[1] Huihui Qiao et al. A time-distributed spatiotemporal feature learning method for machine health monitoring with multi-sensor time series. *Sensors*, 2018.
[2] Junyao Wang et al. Rs2g: Data-driven scene-graph extraction and embedding for robust autonomous perception and scenario understanding. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2024.
[3] Samuel Wilson et al. Hyperdimensional feature fusion for out-of-distribution detection. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2023.
[4] Dequan Wang et al. Tent: Fully test-time adaptation by entropy minimization. In *International Conference on Learning Representations*, 2020.
[5] Han Zhao et al. Multiple source domain adaptation with adversarial learning. In *6th International Conference on Learning Representations, ICLR 2018*, 2018.
[6] Xin Qin et al. Generalizable low-resource activity recognition with diverse and discriminative representation learning. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, page 1943–1953, 2023.
[7] Shiori Sagawa et al. Distributionally robust neural networks. In *International Conference on Learning Representations*, 2019.
[8] Junyao Wang, Luke Chen, and Mohammad Abdullah Al Faruque. Domino: Domain-invariant hyperdimensional classification for multi-sensor time series data. In *2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, pages 1–9. IEEE, 2023.
[9] Yao Qin et al. A dual-stage attention-based recurrent neural network for time series prediction. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, 2017.
[10] Sepp Hochreiter et al. Long short-term memory. *Neural computation*, 1997.
[11] Junyao Wang et al. Disthd: A learner-aware dynamic encoding method for hyperdimensional classification. *arXiv preprint arXiv:2304.05503*, 2023.
[12] Junyao Wang et al. Hyperdetect: A real-time hyperdimensional solution for intrusion detection in iot networks. *IEEE Internet of Things Journal*, 2023.
[13] Mi Zhang et al. Usc-had: A daily activity dataset for ubiquitous activity recognition using wearable sensors. In *Proceedings of the 2012 ACM conference on ubiquitous computing*, 2012.
[14] Ishaan Gulrajani et al. In search of lost domain generalization. In *International Conference on Learning Representations*, 2021.
[15] Qi Dou et al. Domain generalization via model-agnostic learning of semantic features. *Advances in Neural Information Processing Systems*, 2019.
[16] Yaroslav Ganin et al. Domain-adversarial training of neural networks. *The journal of machine learning research*, 2016.
[17] Abbas Rahimi et al. Hyperdimensional biosignal processing: A case study for emg-based hand gesture recognition. In *ICRC*. IEEE, 2016.
[18] Ali Moin et al. A wearable biosensing system with in-sensor adaptive machine learning for hand gesture recognition. *Nature Electronics*, 2021.
[19] Junyao Wang et al. Robust and scalable hyperdimensional computing with brain-like neural adaptations. *arXiv preprint arXiv:2311.07705*, 2023.
[20] Billur Barshan et al. Recognizing daily and sports activities in two open source machine learning environments using body-worn sensor units. *The Computer Journal*, 2014.
[21] Attila Reiss et al. Introducing a new benchmarked dataset for activity monitoring. In *16th international symposium on wearable computers*. IEEE, 2012.
[22] Alejandro Hernández-Cano et al. Onlinehd: Robust, efficient, and single-pass online learning using hyperdimensional system. In *DATE*. IEEE, 2021.