DOMINO: Domain-Invariant Hyperdimensional Classification for Multi-Sensor Time Series Data

Junyao Wang, Luke Chen, Mohammad Abdullah Al Faruque
junyaow4@uci.edu, University of California, Irvine, United States

Motivation

Multi-Sensor Time Series Data:

With the emergence of IoT, heterogeneously connected sensors capture information over time, constituting muti-sensor time series data

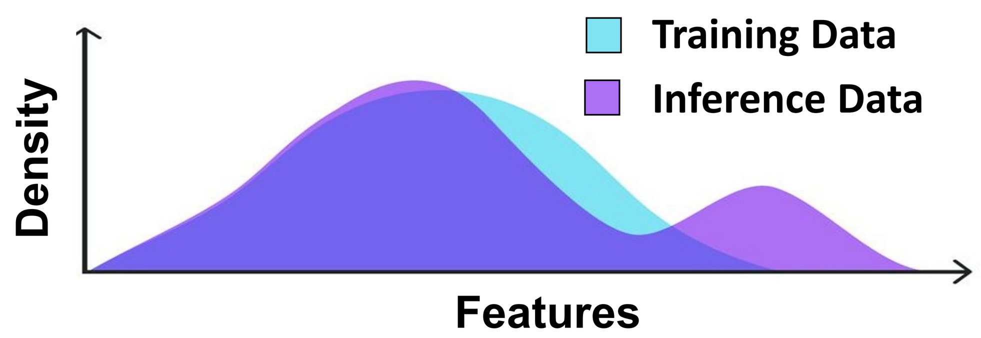
Problem I: Sophisticated DNNs, e.g., RNNs have been proposed to capture spatial and temporal dependencies in these data. ***(Too complicated for edge devices!)***

Problem II: Distribution Shift, a fundamental problem across data-driven ML

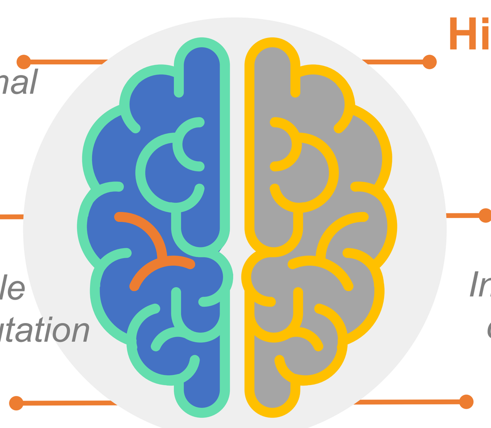
Distribution Shift:

The excellent relies heavily on the critical assumption that the training and inference data are from the same distribution.

Can be easily violated in real-world scenarios and can substantially degrade model performance in many embedded ML applications.



HDC Introduction

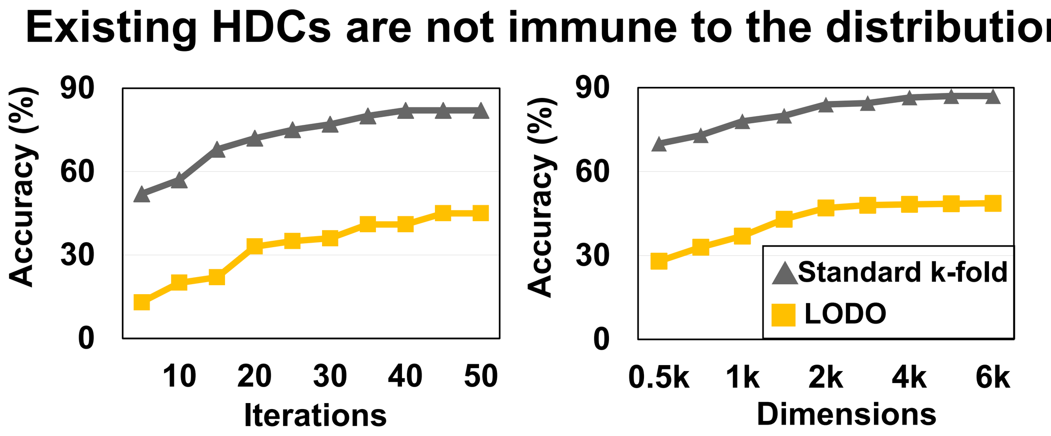


- **Binding (+):** Element-wise addition
  $$\mathcal{H}_{bundle} = \mathcal{H}_1 + \mathcal{H}_2$$
  $$\delta(\mathcal{H}_{bundle}, \mathcal{H}_1) \gg 0, \delta(\mathcal{H}_{bundle}, \mathcal{H}_3) \approx 0$$
- **Bundling:** Element-wise multiplication
  $$\mathcal{H}_{bind} = \mathcal{H}_1 * \mathcal{H}_2$$
  $$\delta(\mathcal{H}_{bind}, \mathcal{H}_1) \approx 0, \delta(\mathcal{H}_{bind}, \mathcal{H}_2) \approx 0$$
- **Reasoning:** measuring the similarity of hypervectors, e.g. cosine similarity is calculated as $\delta(\mathcal{H}_1, \mathcal{H}_2) = \frac{\mathcal{H}_1 \cdot \mathcal{H}_2}{\|\mathcal{H}_1\| \cdot \|\mathcal{H}_2\|}$

- ***A powerful learning solution for today's edge platforms***
  - ✓Fast convergence, high computational efficiency, ultra-robustness against noise
  - ✓ High-quality results comparable to SOTA DNNs
- ***Incorporates learning capability along with storing/loading information***
  - ✓ Unique advantages in dealing with time-series data

Challenges

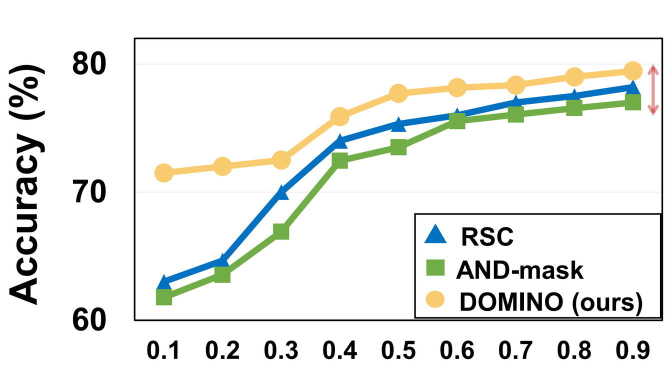
Existing HDCs are not immune to the distribution shift issue



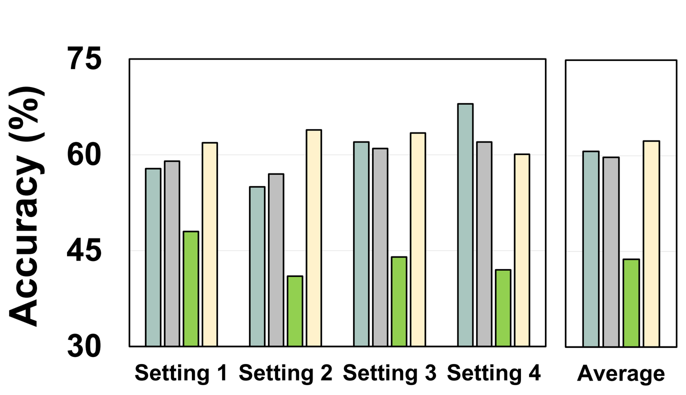
The accuracy of leave-one-domain-out (LODO) cross-validation (CV) is considerably lower than the standard k-fold CV.

A very limited generalization capability of existing models.

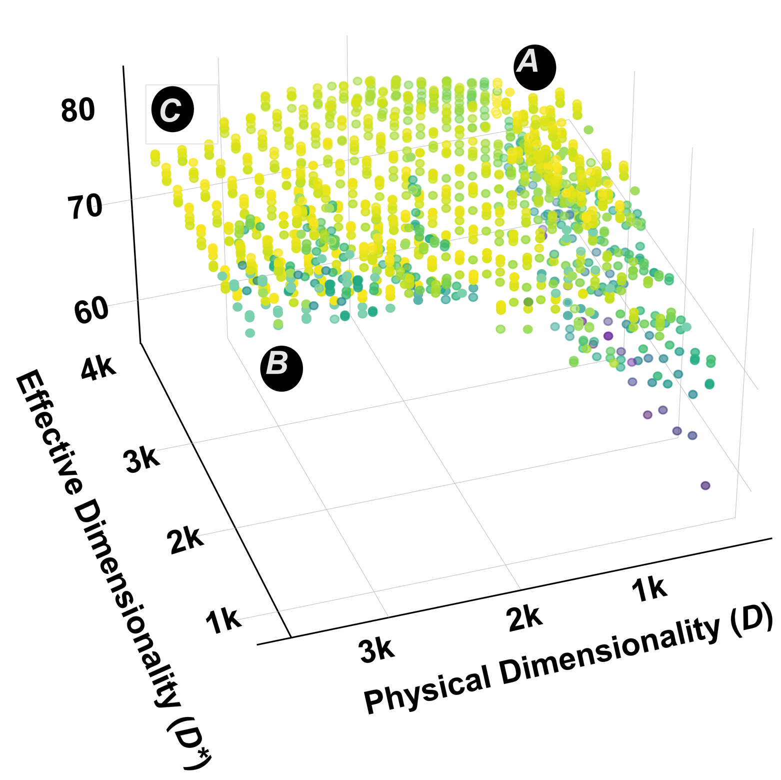
Evaluations



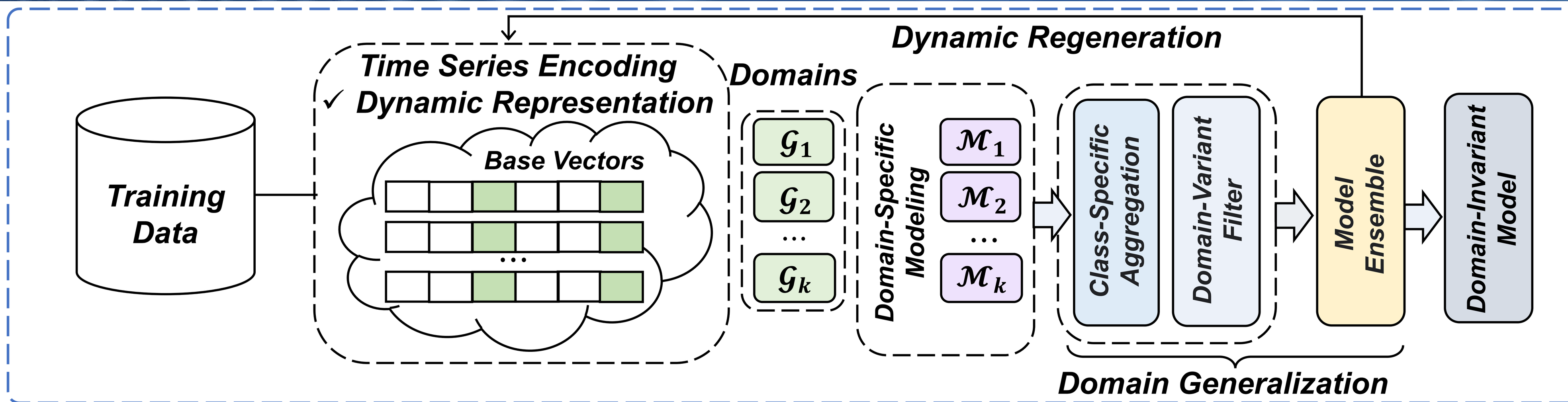
When using only 10% of the training data, DOMINO demonstrates 5.81% higher accuracy than AND-mask and 4.90% high accuracy than RSC.



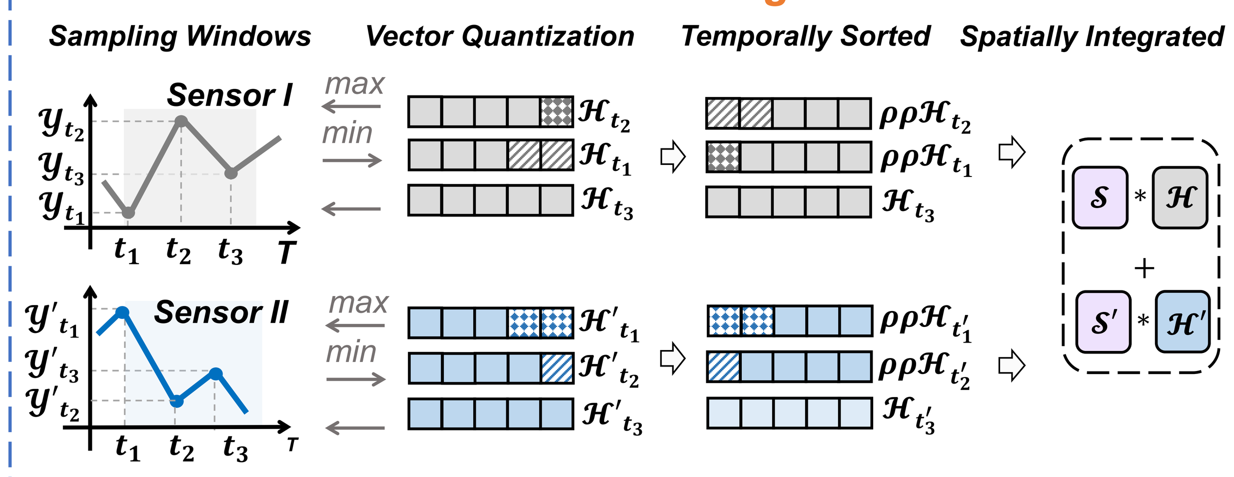
DOMINO outperforms SOTA CNN-based DG approaches by exhibiting on average 1.18% and 2.58% higher accuracy than RSC and AND-mask.



DOMINO: HDC-Based Domain Generalization



Multi-Sensor Time Series Encoding



N-gram Encoding:

- Assign random hypervectors $\mathcal{H}_{max}$ and $\mathcal{H}_{min}$ to represent the maximum and minimum signal values, i.e., y_{max} and y_{min}.
- Vector quantization to values between y_{max} and y_{min}. For instance,

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

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

- Temporally sorting by rotation shifts (ρ), e.g., $\mathcal{H} = \rho\rho\mathcal{H}_{t_1} * \rho\mathcal{H}_{t_2} * \mathcal{H}_{t_3}$
- Spatially integrating by binding, e.g., $\mathcal{H} = \mathcal{S}_1 * \mathcal{H}_1 + \cdots + \mathcal{S}_n * \mathcal{H}_n$

Domain-Specific Modeling

Algorithm 1 Domain-Specific Modeling

Input: $\mathcal{N}$ encoded training samples $\mathcal{H}_1, \mathcal{H}_2, \ldots, \mathcal{H}_\mathcal{N}$ with n classes and k domains, domains for training data $\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_k$, label of each data sample $\mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_\mathcal{N}$, learning rate η.
Output: k domain-specific models $\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_k$ after one training iteration.

```
for each λ ∈ [1, k] do
    Initialize a domain-specific model M_λ consisting of one class
    hypervectors for each class M_λ = {C_λ^1, C_λ^2, ..., C_λ^n}
    for each H_i ∈ {H_1, H_2, ..., H_N} do
        if domain(H_i) = G_λ then
            C_max = arg max_{C_λ^t} {δ(H_i, C_λ^1), ..., δ(H_i, C_λ^n)}
            if L_i = C_max then
                continue
            else if L_i ≠ C_max ∧ L_i = C_λ^j then
                C_max ← C_max − η · [1 − δ(H_i, C_max)] × H_i
                C_λ^j ← C_λ^j + η · [1 − δ(H_i, C_λ^j)] × H_i
    M_λ = Normalize(M_λ)
return M_1, M_2, ..., M_k
```

Our algorithm provides a higher chance for non-common patterns to be properly included

$$\delta(\mathcal{H}, \mathcal{C}_\lambda^t) = \frac{\mathcal{H} \cdot \mathcal{C}_\lambda^t}{\|\mathcal{H}\| \cdot \|\mathcal{C}_\lambda^t\|} = \frac{\mathcal{H}}{\|\mathcal{H}\|} \cdot \frac{\mathcal{C}_\lambda^t}{\|\mathcal{C}_\lambda^t\|}$$

- *A large $\delta(\mathcal{H}, \cdot)$ indicates the input data points is marginally mismatches*
- *A small $\delta(\mathcal{H}, \cdot)$ indicates a noticeably new pattern*

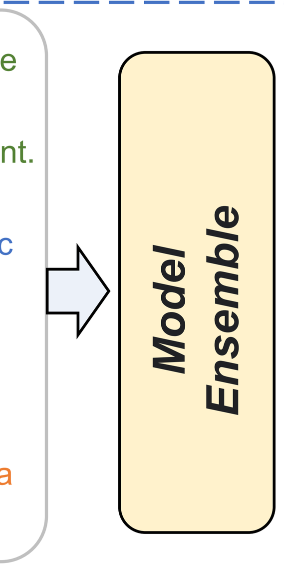
Domain Generalization (DG)

Algorithm 2 Domain Generalization

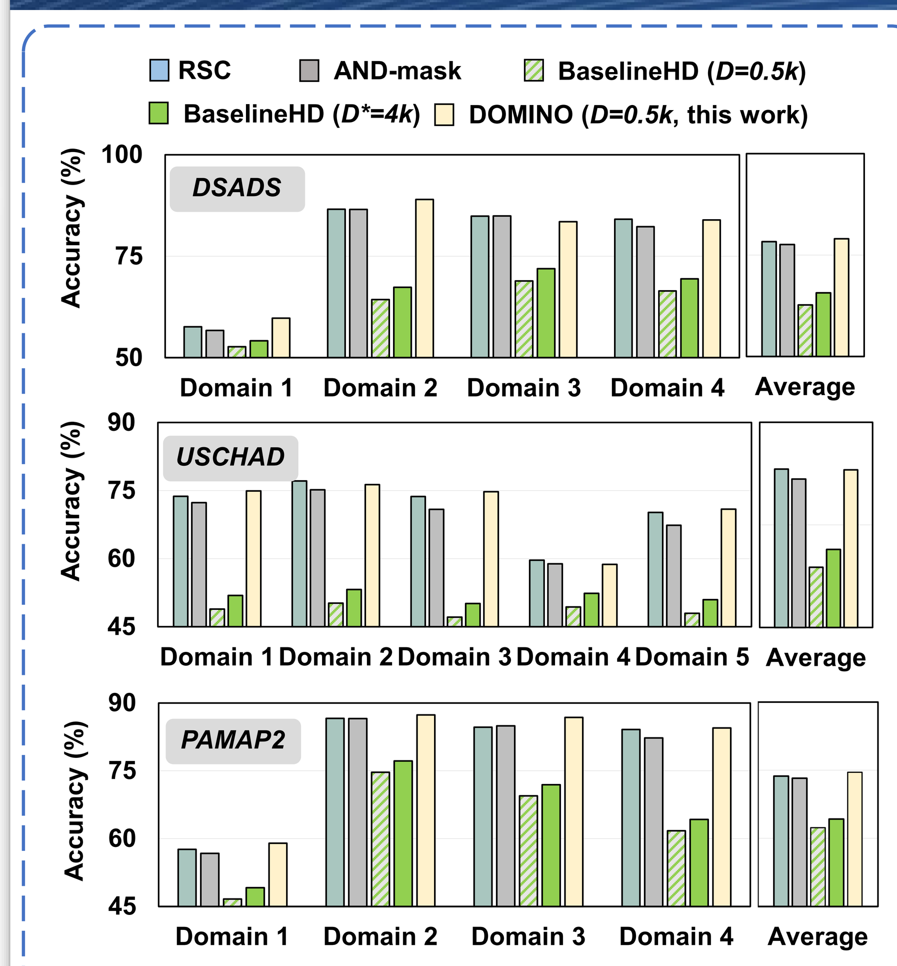
Input: k domain-specific models $\{\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_k\}$ each with size $n \times \mathcal{D}$, regeneration rate $\mathcal{R}$.
Output: Domain-variant dimensions $\mathcal{U}$ to drop.

```
Initialize n empty matrices T_1, T_2, ..., T_n, each with size k × D.
for each T_γ ∈ {T_1, T_2, ..., T_n} do
    for each M_λ ∈ {M_1, M_2, ..., M_k} do
        T_γ[λ, :] = M_λ[γ, :]          ▷ Form n class-specific matrices
    σ_γ = Variance(T_γ, columnwise)     ▷ dim(σ_γ) = 1 × D
V = Σ_{i=1}^{n} σ_i
U = argsort(V)[⌊(1 − R) · D⌋ : D]     ▷ R of dimensions with largest variance
return U
```

- Dimensions with large variance indicate, for the same class, these dimensions store very different information, and are hence considered domain-variant.
- We sum up the variance vector of each class-specific matrix to obtain a vector representing the overall relevance of dimension to domains.
- We select the top $\mathcal{R}$ portion of dimensions with the highest variance and **regenerate** each of them with a new randomly generated vector.



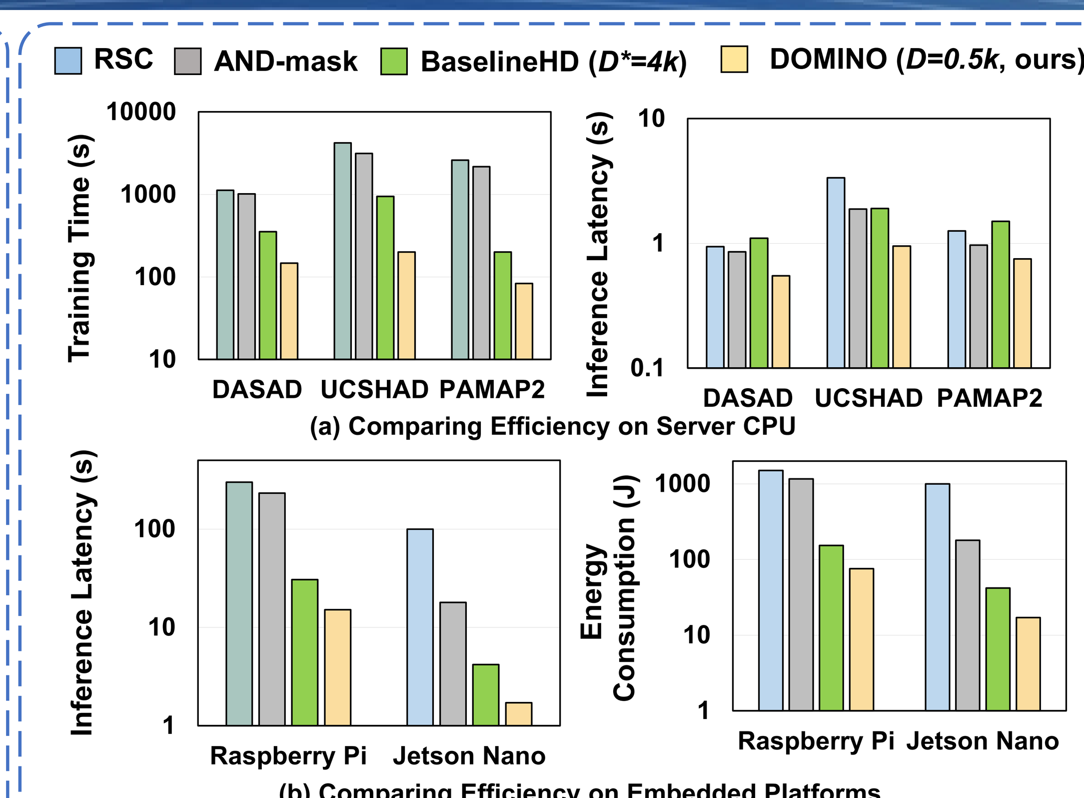
Evaluations



LODO Accuracy:

0.96% higher than RSC, 2.04% higher than AND-mask

11.70% higher than BaselineHD ($\mathcal{D}^* = 4k$), 15.93% higher than BaselineHD ($\mathcal{D} = 0.5k$)



On Server CPU, DOMINO exhibits:

- 16.34× faster training than RSC, 14.17× faster training than AND-mask
- 2.89 × faster inference than RSC, 1.97 × faster inference than AND-mask

On embedded devices:

- Raspberry Pi:
  - ✓19.79 × faster than RSC, 15.31× faster than AND-mask
- Jetson Nano
  - ✓58.44 × faster than RSC, 10.49 × faster than AND-mask

Conclusion:

We propose DOMINO, an HDC-based domain generalization algorithm that provides significantly higher efficiency and better performance than SOTA DNN-based techniques. Our solution provides a resource-efficient and hardware-friendly solution, especially for today's edge devices, to mitigate the distribution shift challenge.