RESOURCE-EFFICIENT FEDERATED HYPERDIMENSIONAL COMPUTING

Nikita Zeulin [1] **Olga Galinina** [1 2] **Nageen Himayat** [3] **Sergey Andreev** [1]

ABSTRACT

In conventional federated hyperdimensional computing (HDC), training larger models usually results in higher predictive performance but also requires more computational, communication, and energy resources. If the system resources are limited, one may have to sacrifice the predictive performance by reducing the size of the HDC model. The proposed resource-efficient federated hyperdimensional computing (RE-FHDC) framework alleviates such constraints by training multiple smaller independent HDC sub-models and refining the concatenated HDC model using the proposed dropout-inspired procedure. Our numerical comparison demonstrates that the proposed framework achieves a comparable or higher predictive performance while consuming less computational and wireless resources than the baseline federated HDC implementation.

1 INTRODUCTION

1.1 Motivation

Developing efficient frameworks that mitigate the effects of computational, communication, and data heterogeneity in mobile edge networks remains an open challenge in federated learning research (Kairouz et al., 2021). The superior performance of artificial neural networks (ANNs) in many machine learning (ML)-aided applications motivates the development of more efficient federated methods that may decrease computational, communications, and energy costs induced by the overparametrization of ANNs. Some of these methods include binary ANNs (Kim & Smaragdis, 2016), gradient compression (Bernstein et al., 2018), or user subsampling (Nguyen et al., 2020).

Recently, the research community has shown increasing interest in hyperdimensional computing (HDC) (Kanerva, 2009), a hardware-efficient ML approach that promises to become a compact and low-complexity alternative to ANN-based models. The key difference between HDC and the conventional ML models is in more hardware-efficient implementation of the training and inference procedures. In HDC, all the data are mapped to randomized, very large, hyperdimensional (HD) binary, bipolar, or real-valued vectors. Operations over HD vectors can be efficiently implemented in hardware using low-cost bitshifts and exclusive ORs. In high-dimensional spaces, the HD vectors of one class can be aggregated or bundled into a so-called HD prototype preserving similarity with the bundled data. Owing to this property, the inference procedure reduces to computing Hamming or cosine distances between the HD prototypes, which significantly boosts the computational and energy efficiency as compared to matrix-to-matrix multiplications employed in ANNs. To date, HDC has been adapted to classification of images (Dutta et al., 2022), time series (Schlegel et al., 2022), graphs (Nunes et al., 2022), and text (Kanerva, 2009) as well as to regression problems (Hernández-Cano et al., 2021) and others. For a detailed discussion on HDC architectures and their applications, we refer to Kleyko et al. (2023a) and Kleyko et al. (2023b).

1.2 Related Work

There are several federated HDC implementations that have been proposed recently. The work in Hsieh et al. (2021) introduced a procedure for federated training of HDC models. The main idea of that method is to utilize bipolar HDC encoding to reduce the amount of transmitted information in the number of bits as compared to conventional ML models with real-valued parameters. A similar principle was exploited in Chandrasekaran et al. (2022) with the main difference that the considered HDC model adopted a convolutional neural network (CNN)-based feature extractor, the use of which has been shown to considerably enhance the performance of HDC in image classification problems (Dutta et al., 2022). The discussion in Zhao et al. (2022) highlighted that using a straightforward federated model averaging can lead to performance degradation and, therefore, it was suggested using a weighted average of the local and global models to avoid a performance drop. A federated decentralized HDC approach for training randomized neural

[1]Tampere University, Tampere, Finland [2]Tampere Institute for Advanced Study, Tampere, Finland [3]Intel Corporation, Santa Clara, CA, USA. Correspondence to: Nikita Zeulin <nikita.zeulin@tuni.fi>.

Proceedings of the 6th MLSys Conference Workshop on Federated Learning Systems, Miami, FL, USA, 2023. Copyright 2023 by the author(s).

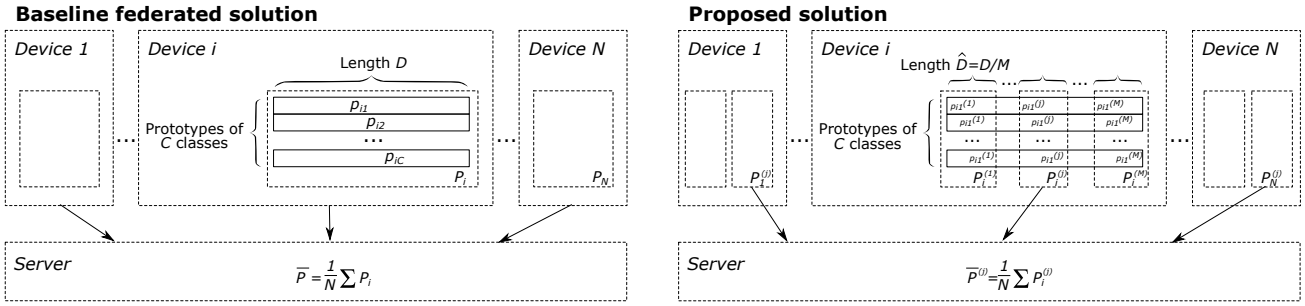

Figure 1. Comparison of proposed RE-FHDC solution and baseline federated HDC.

networks was proposed in Diao et al. (2021), where HDC was applied as a more computationally-efficient alternative to optimizing the output layer using ordinary least squares. In that method, the properties of HDC were also exploited to compress the transmitted HDC model into a single HD vector, thus reducing communication overheads.

In general, the existing federated HDC implementations achieve computational and communication efficiency improvements by utilizing binary or bipolar HD representations. In this paper, we show that one can exploit the properties of HDC to further reduce the computational and communication costs of federated training. Particularly, we propose a resource-efficient federated HDC method named RE-FHDC, which reduces computational and communication costs per a single federated learning round by partitioning a larger HDC model into independently trained HDC sub-models and further concatenating them into a single one after federated training. Below, we outline the adopted HDC model and describe the principle of our RE-FHDC.

2 Federated HDC

In the considered system model, N user devices collaboratively train a real-valued D-dimensional HDC model, which has a set of prototypes $\{\mathbf{p}_i\}_{i=1}^{C}$ corresponding to each of C predicted classes. We further refer to the HDC model as a real-valued $C \times D$ matrix $\mathbf{P} = [\mathbf{p}_1, \ldots, \mathbf{p}_C]$ formed of the HD prototypes.

The HDC model training includes three successive procedures: (i) data transform, (ii) prototype initialization, and (iii) prototype retraining. During the data transform procedure, d-dimensional training data are mapped to HD vectors using the selected HDC mapping $\theta : \mathbb{R}^d \to \mathbb{R}^D$. Then, the computed HD representations of the data are bundled (we employ element-wise summation) into C HD prototypes of the corresponding classes as $\mathbf{p}_i = \sum_{\mathbf{x} \in \mathcal{C}_i} \theta(\mathbf{x})$, where $\mathcal{C}_i$ is a subset of data corresponding to the i-th class. The inference procedure in HDC includes computing a similarity measure or distance $\mathrm{dist}(\theta(\mathbf{x}^*), \mathbf{p}_i)$ between the HD representation $\theta(\mathbf{x}^*)$ of the test data point $\mathbf{x}^*$ and each prototype $\mathbf{p}_i$ and then selecting a class with the minimum distance. The formed prototypes can be iteratively refined by running several iterations of inference over the training data. If incorrect classifications occur, the prototypes are updated using a method-specific update rule.

The existing HDC frameworks differ primarily in the specific implementation of the introduced procedures. Below, we discuss the implementation of these procedures in our RE-FHDC solution.

2.1 Random Projection-Based Data Transform

Any selected HDC mapping should have three essential properties: representation distributiveness, similarity preservation, and implementation efficiency. The first property guarantees that an aggregate of the HD vectors, or the HD prototype, is similar to each of the aggregated HD vectors, which is most commonly achieved by employing randomized mappings and feature expansions. The second property ensures that the HDC mapping preserves similarity of the original d-dimensional data in the HD space, which improves the robustness of inference over the transformed HD data. The third property implies that the selected mapping θ and the similarity measure can be efficiently implemented in hardware.

In the RE-FHDC method, we adopt a random projection-based mapping as proposed in the OnlineHD framework (Hernandez-Cane et al., 2021):

$$\theta(\mathbf{x}) = \cos(\mathbf{x}\mathbf{W} + \boldsymbol{\varphi}) \cdot \sin(\mathbf{x}\mathbf{W}), \tag{1}$$

where $\mathbf{x}$ is a d-dimensional data point, $\mathbf{W} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ is a $d \times D$-dimensional random projection matrix, and $\boldsymbol{\varphi} \sim \mathrm{Uni}[0, 2\pi]$ is a D-dimensional random vector. The generated random projection parameters $\mathbf{W}$ and $\boldsymbol{\varphi}$ are non-learnable and remain fixed throughout the training procedure. For the employed random projection-based mapping, we use cosine distance $\mathrm{dist}(\mathbf{a}, \mathbf{b}) = 1 - \frac{\langle \mathbf{a}, \mathbf{b} \rangle}{\|\mathbf{a}\|_2 \|\mathbf{b}\|_2}$ to measure the similarity between two HD vectors $\mathbf{a}$ and $\mathbf{b}$.

The employed random projection-based mapping has several advantages over more conventional algebraic HDC implementations, such as holographic reduced representations (Plate, 1995) or multiply-accumulate-permute (Ge & Parhi, 2020; Kanerva, 2009). First, multiple data points can be transformed in a single matrix-to-matrix multiplication operation, which is parallelized using dedicated libraries, such as OpenBLAS, OpenCL, and CUDA, with minimal user intervention. In contrast, algebraic implementations require individual processing of each data point, while its efficient parallelization has tp be implemented separately (Kang et al., 2022a;b). Second, matrix-to-matrix multiplications can be efficiently performed on GPUs or dedicated AI chips of mobile systems-on-chips with lower energy costs as compared to the CPU-based processing. To achieve the promoted reduction in energy costs and processing times, algebraic HDC requires low-level hardware-specific manipulations, as demonstrated for in-memory (Karunaratne et al., 2020) and FPGA-based (Imani et al., 2021) HDC implementations.

Based on this discussion, we find the random projection-based HDC transform to be a more convenient and universal option for general-purpose mobile platforms, which is the reason for preferring it in our federated HDC framework.

2.2 Prototype Construction and Retraining

In our RE-FHDC solution, the prototypes of each of C classes are constructed by summing the HD representations of the corresponding data into a single prototype $\mathbf{p}_i = \sum_{\mathbf{x} \in \mathcal{C}_i} \alpha \cdot \theta(\mathbf{x})$, where $\alpha \in [0, 1]$ is the learning rate. In the dataset, some data points may have high similarity but belong to different classes, which makes the corresponding class prototypes "fuzzy". The goal of the subsequent retraining procedure is to increase dissimilarity between the prototypes by reinforcing the correct predictions and penalizing the incorrect ones by using the iterative refining procedure from the OnlineHD algorithm (Hernandez-Cane et al., 2021). If the training data point $\mathbf{x}$ of class i is misclassified into class j, then the corresponding HD prototypes $\mathbf{p}_i$ and $\mathbf{p}_j$ are updated as:

$$\mathbf{p}_i = \mathbf{p}_i - \alpha \cdot (1 - \Delta_i) \cdot \theta(\mathbf{x}), \ \mathbf{p}_j = \mathbf{p}_j + \alpha \cdot (1 - \Delta_j) \cdot \theta(\mathbf{x}), \tag{2}$$

where Δ_i and Δ_j are the cosine distances between the transformed data point $\theta(\mathbf{x})$ and the prototypes $\mathbf{p}_i$ and $\mathbf{p}_j$, respectively.

2.3 Federated Training

The employed federated training setup follows the FedAvg algorithm (Konečnỳ et al., 2016) and includes G global epochs. During each global epoch, the devices perform L local epochs of HDC model retraining as in (2) and transmit their updated local HDC models to the parameter server. At the end of each global epoch, the parameter server aggregates the prototypes and returns the averaged HDC model to the user devices:

$$\bar{\mathbf{P}} = \frac{1}{N} \sum_{j=1}^{N} \mathbf{P}_j, \tag{3}$$

where $\mathbf{P}_j$ is a local HDC model of the j-th participant. We explicitly note that the participants should use identical HDC mapping θ, which can be achieved, for example, by sharing a common seed for the random number generators.

As discussed in Zhao et al. (2022), adopting the model averaging in (3) can experience performance degradation and, therefore, the local HDC models of the participants $\mathbf{P}_j$ should be weighed with the received average $\bar{\mathbf{P}}$. In our results, we do not observe any noticeable reduction of performance degradation after adopting the model update rule from Zhao et al. (2022). Instead, reducing the number of local iterations L and tuning the HDC retraining parameters can stabilize the federated training of HDC models, especially for lower dimensionalities D. We note that a performance degradation can be successfully avoided with appropriate selection of the federated training hyperparameters.

3 Proposed RE-FHDC Method

In the existing federated HDC solutions, the devices train full-sized D-dimensional HDC models throughout the entire federated training process. While opting for larger values of D generally leads to higher predictive performance, it also results in increased communication overheads and local training times due to several reasons. First, the refining procedure in (2) involves repeated similarity checks with a complexity of $O(CD)$, thus having longer processing times for larger D. Second, simultaneous transmissions of large model updates from multiple (in practice, thousands) devices faces a communication bottleneck, where the time required to receive all the model updates becomes considerably higher than that needed to compute the model updates themselves (Kairouz et al., 2021). Third, larger sizes D of the HDC models require more available RAM/VRAM for storing the HD representations and the results of intermediate computations. Even though batched data processing can alleviate this limitation, it may introduce additional computational overheads by repeating the HDC mapping during the retraining procedure.

3.1 Core Idea of Proposed Solution

The proposed RE-FHDC method overcomes the aforementioned limitations by collaboratively training M independent HDC sub-models $\{\mathbf{P}^{(1)}, \ldots, \mathbf{P}^{(M)}\}$ of dimensionality $\hat{D} = D/M$ and performing inference with the concatenated

HDC model $\mathbf{P}_C = [\mathbf{P}^{(1)}, \ldots, \mathbf{P}^{(M)}]$ (see Fig. 1). Our RE-FHDC method comprises two successive stages: federated training and federated refining, which have the duration of G_T and G_R global epochs, respectively.

The federated training stage is divided into M sub-stages of G_T/M global epochs, where the participants sequentially train M $\hat{D}$-dimensional HDC sub-models as described in Section 2.3. That is, by the end of the G_T-th global epoch, the participants have M collaboratively trained $\hat{D}$-dimensional HDC sub-models. Upon completion of the federated training procedure, the participants concatenate M $\hat{D}$-dimensional HDC sub-models into a single D-dimensional HDC model $\mathbf{P}_C = [\mathbf{P}^{(1)} \ldots \mathbf{P}^{(M)}]$ and can then perform the inference by using the concatenated HDC model $\mathbf{P}_C$.

During the federated refining stage that follows after G_T global epochs, the participants randomly select a subset of D_0 HD prototype positions and perform one global epoch of federated training over these randomly selected positions. That is, at each global epoch, the participants update only a subset of positions of the concatenated HDC model $\mathbf{P}_C$. While the hyperparameters (G_T, G_R) may be customized for a particular task, we observe that the best practice is to set $G_T = M$, i.e., to perform a single global iteration for each HDC sub-model and allocate the remaining global iterations for federated refining. Even though the refining stage is optional, we observe that it is the key feature of RE-FHDC that considerably improves the predictive performance when compared to the baseline while requiring lower training costs, as demonstrated further in Section 4.

3.2 Complexity Analysis

Let us compare the computational and communication complexities of the baseline federated HDC and the proposed RE-FHDC solutions. We assume that the computational complexity of the random projection-based data transform in (1) is determined by the matrix-to-matrix multiplication of $\mathcal{C}_1(D) = |\mathbf{X}| \cdot D \cdot (2d+1)$ floating-point operations, where $|\mathbf{X}|$ is the number of transformed original data points. After the original data are projected onto the HD space, the HD representations are summed element-wise to construct the HD prototypes: the total computational complexity of this operation is $\mathcal{C}_2(D) = |\mathbf{X}| \cdot D$. The retraining procedure involves computing pairwise distances between the HD representations and C HD prototypes with further refining of the prototypes over the misclassified data points. We assume that the computational complexity of one epoch of the retraining procedure is determined by the complexity of computing pairwise distances and equals $\mathcal{C}_3(D) = |\mathbf{X}| \cdot C \cdot 3(2D+1)$.

Based on the above discussion, the computational complexity of the baseline federated HDC is $\mathcal{C}_B(D) = \mathcal{C}_1(D) + \mathcal{C}_2(D) + L \cdot G \cdot \mathcal{C}_3(D)$. In our RE-FHDC method, each of M HDC sub-models of dimensionality $\hat{D} = D/M$ is independently trained during G_T/M global epochs. This procedure is followed by the retraining over the subsets of D_0 randomly selected positions of the concatenated HDC model during G_R global epochs. Therefore, the total computational complexity of our RE-FHDC method is

$$\begin{aligned}\mathcal{C}_R(D) &= M\,[\mathcal{C}_1(\hat{D}) + \mathcal{C}_2(\hat{D})] + L\,[G_T\,\mathcal{C}_3(\hat{D}) + G_R\,\mathcal{C}_3(D_0)] \\ &= \mathcal{C}_1(D) + \mathcal{C}_2(D) + L\,[G_T\,\mathcal{C}_3(\hat{D}) + G_R\,\mathcal{C}_3(D_0)],\end{aligned}$$

which is identical to that of the baseline federated HDC for $\hat{D} = D$, $G_R = 0$, and $G_T = G$. That is, one can reduce the total computational and communication costs by selecting larger numbers of HDC sub-models M and smaller values of D_0. The resulting slower convergence in terms of iterations can be compensated for by lower training costs, as we demonstrate in Section 4.

3.3 Discussion

Below, we briefly describe the intuition behind the proposed RE-FHDC method by leaving a rigorous theoretical analysis for future work. The introduced federated training procedure leverages the independence of the elements of the random projection matrix $\mathbf{W}$ and the random vector $\boldsymbol{\varphi}$. One can readily show that if $\mathbf{x}_1$ and $\mathbf{x}_2$ are arbitrary d-dimensional vectors, while the distance $\mathrm{dist}(\theta(\mathbf{x}_1), \theta(\mathbf{x}_2))$ and the random projection-based mapping $\theta(\mathbf{x})$ are defined as given in Section 2.1, then $\mathrm{dist}(\theta(\mathbf{x}_1), \theta(\mathbf{x}_2))$ is a Monte-Carlo estimate of its expectation, and

$$\lim_{D \to \infty} \mathrm{dist}(\theta(\mathbf{x}_1), \theta(\mathbf{x}_2)) = \mathbb{E}_{(\mathbf{W}, \boldsymbol{\varphi})}[\mathrm{dist}(\theta(\mathbf{x}_1), \theta(\mathbf{x}_2))]. \tag{4}$$

Essentially, the size $\hat{D}$ of the HDC sub-model determines the variance of the distance estimate in (4). If $\hat{D}$ is sufficiently large, then the elements of the HDC sub-models $\{\mathbf{P}^{(1)}, \ldots, \mathbf{P}^{(M)}\}$ have similar probability distributions, and this similarity can be facilitated by fixing the order of the training data during the local retraining procedure. We also observe that there is a practical lower limit on the value of D_0, after which our RE-FHDC method fails to converge. This discussion yields the following Proposition.

Proposition 3.1. *If the size $\hat{D}$ of the HDC sub-model is sufficiently large, then the concatenation $\mathbf{P}_C = [\mathbf{P}^{(1)}, \ldots, \mathbf{P}^{(M)}]$ of M independently trained HDC sub-models has a comparable or higher performance to that of the HDC model of size $D = \hat{D}M$.*

The underlying principle of the federated refining is similar to the well-known dropout method (Srivastava et al., 2014) for ANN training, where the inter-layer connections are dropped randomly to reduce model overfitting. Using a smaller subset of D_0 randomly selected positions to estimate the distance in (4) increases the estimation variance and,

Table 1. Comparison of methods in terms of maximum achieved accuracy, i.i.d. setup.

	Maximum achieved accuracy						
	RE-FHDC, $D = 5$K			Baseline federated HDC			
	$\hat{D} = 2.5$K	$\hat{D} = 1$K	$\hat{D} = 0.5$K	$D = 5$K	$D = 2.5$K	$D = 1$K	$D = 0.5$K
MNIST	**0.972**	**0.969**	**0.963**	0.965	0.959	0.940	0.913
Fashion MNIST	**0.883**	**0.873**	**0.862**	0.877	0.867	0.852	0.840
CIFAR-10	**0.492**	**0.484**	**0.447**	0.499	0.477	0.440	0.417
UCI HAR	**0.944**	**0.945**	**0.945**	0.936	0.932	0.915	0.904
Size of trained HDC model	100KB	40KB	20KB	200KB	100KB	40KB	20KB

Table 2. Comparison of methods in terms of rounds to converge and total uplink traffic, i.i.d. setup.

	Rounds to reach baseline $D = 5$K				Total uplink traffic, MB			
	Baseline, $D = 5$K	RE-FHDC, $\hat{D}$			Baseline, $D = 5$K	RE-FHDC, $\hat{D}$		
		2.5K	1K	0.5K		2.5K	1K	0.5K
MNIST	30	27	60	100+	240	**108 (−55%)**	**96 (−60%)**	80+
Fashion MNIST	40	60	100	100+	320	**240 (−25%)**	**160 (−50%)**	80+
CIFAR-10	5	18	26	100+	**40**	72 (+80%)	41 (+2.5%)	80+
UCI HAR	20	22	35	50	160	**88 (−45%)**	**56 (−65%)**	**40 (−75%)**

therefore, the likelihood of incorrect classification, which leads to changing the prototypes during the local retraining procedure in (2). In this case, the HDC model can escape a local minimum and continue its convergence to a better solution. Therefore, one can consider this refining procedure as a regularization method to mitigate overfitting in random projection-based HDC.

4 Numerical Results

We compare our RE-FHDC method to the baseline federated HDC implementation. The considered baseline differs from the proposed method in two key aspects: (i) the participants collaboratively train a full-sized D-dimensional HDC model and (ii) the participants do not refine the global model as proposed in our RE-FHDC solution. This approach is based on the conventional federated learning principles, where the participants upload the entire updated model, and has been adopted by the existing HDC implementations (Hsieh et al., 2021; Chandrasekaran et al., 2022; Zhao et al., 2022; Diao et al., 2021).

4.1 Experimental Setup

We compare our RE-FHDC solution to the baseline federated HDC over several datasets: MNIST (LeCun, 1998), Fashion MNIST (Xiao et al., 2017), UCI HAR (Anguita et al., 2013), and CIFAR-10 (Krizhevsky et al., 2009) under the i.i.d. and non-i.i.d. setups. In the i.i.d. scenario, the participants have identical or near-identical class distributions of the training data, while in the non-i.i.d. scenario, each participant has the training data corresponding to two randomly selected distinct classes.

We additionally employ random Fourier feature mapping (RFFM) (Rahimi & Recht, 2007), which is a feature transform projecting the data onto a higher-dimensional space to facilitate their linear separability. The results in Yu et al. (2022); Yan et al. (2023) demonstrate that linear separability is particularly important for HDC performance, especially in image classification tasks (Dutta et al., 2022). The radial basis function kernel most commonly approximated with RFFM has been widely used beyond image classification, which makes RFFM a universal solution for different types of data. The proposed RE-FHDC method is compatible with arbitrary feature extractors, including the state-of-the-art ANN-based options as in Dutta et al. (2022), and, hence, its feature extraction stage is largely implementation-specific. We set the number of RFFM features to 3.2K and the RFFM length-scale parameter to $\sigma = 1$ for the image datasets and to $\sigma = 2.5$ for the UCI HAR dataset.

4.2 Discussion of Key Results

We compare the baseline and the proposed HDC solutions for $N = 20$ participants. We set the size of the HDC model to $D = 5$K, $D_0 = \hat{D}$, and vary the number of HDC sub-models as $M = [10, 5, 2]$. We additionally assess the performance of the baseline federated HDC model of dimensionality D/M, which implies the same training costs as those in the proposed solution. We set the total number of global epochs to $G = 100$ and the number of local epochs to $L = 5$ and $L = 3$ for the i.i.d. and the non-i.i.d. scenarios, respectively.

In Table 1, we report the maximum accuracy achieved by the considered federated HDC methods under the i.i.d. setup

Table 3. Comparison of methods in terms of maximum achieved accuracy, non-i.i.d. setup.

	RE-FHDC, $D = 5$K			Baseline federated HDC			
	$\hat{D} = 2.5$K	$\hat{D} = 1$K	$\hat{D} = 0.5$K	$D = 5$K	$D = 2.5$K	$D = 1$K	$D = 0.5$K
MNIST	**0.926**	**0.924**	**0.919**	**0.927**	0.921	0.905	0.879
Fashion MNIST	**0.778**	**0.775**	**0.773**	**0.779**	0.776	0.774	0.762
CIFAR-10	**0.363**	**0.359**	**0.355**	0.363	0.359	0.352	0.345
UCI HAR	**0.922**	**0.901**	0.861	0.898	0.888	0.876	**0.862**
Size of trained HDC model	100KB	40KB	20KB	200KB	100KB	40KB	20KB

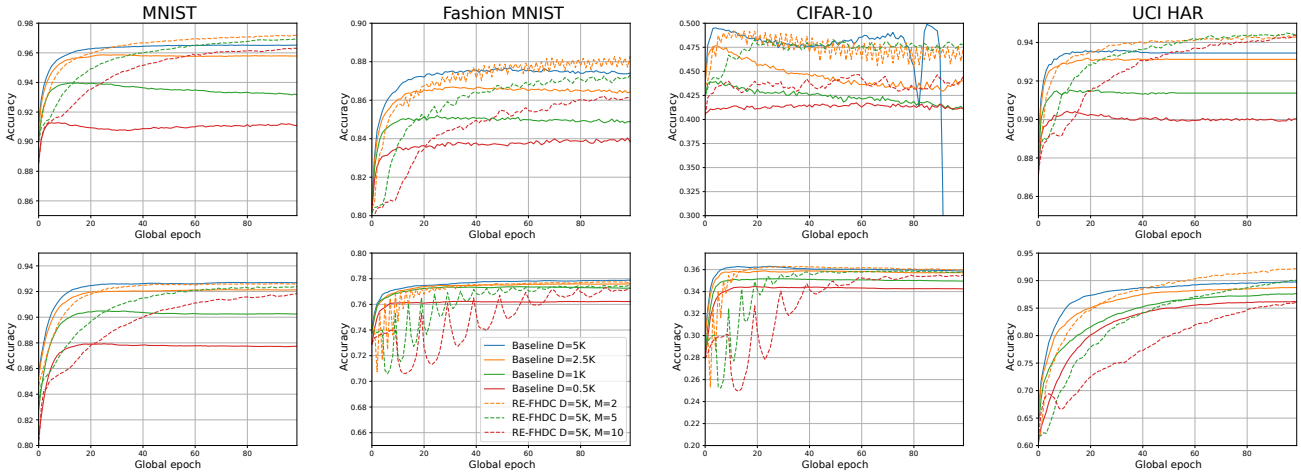

Figure 2. Performance comparison of RE-FHDC and baseline federated HDC under i.i.d. (top) and non-i.i.d. (bottom) setups.

after G global iterations. One may see that the proposed RE-FHDC method achieves a comparable or higher accuracy than that of the baseline, while processing smaller HDC models reduces the computational and communication costs at each global epoch. In Table 2, we demonstrate the total number of global epochs required to converge to the accuracy of the baseline federated HDC model with $D = 5$K as well as the corresponding total volume of data transmitted by the participants. The results suggest that our RE-FHDC method can reduce the network traffic by up to $\times 4$ times without sacrificing the predictive performance. A comparison of the maximum accuracy of the methods under the non-i.i.d. setup is provided in Table 3.

In Fig. 2, we compare the accuracy evolution of our RE-FHDC vs. the baseline federated HDC method. We hypothesize that the observed instability in the i.i.d scenario is mainly due to an imperfect selection of the feature extractor. The work in Dutta et al. (2022); Chandrasekaran et al. (2022) demonstrates a significantly smoother convergence with a CNN-based feature extractor, and, therefore, we argue that our RE-FHDC method can be further enhanced by applying more advanced data preprocessing techniques. In the non-i.i.d. scenario, we observe a noticeably smoother performance evolution, which may be attributed to a more homogeneous distribution of the local data. Since the participants only store the data of two classes, the local retraining procedure yields smaller changes in the local prototypes of the absent classes, thus increasing stability of the federated training. For the more complex datasets, such as Fashion MNIST and CIFAR-10, we observe periodicity in the accuracy evolution with peaks at every M-th iteration. The reason for this is that the accuracy of inference is measured with the *concatenated* HDC model, and by the M-th iteration, all of D positions of the HDC model become updated, resulting in an accuracy leap. We do not observe such behavior for MNIST and UCI HAR datasets presumably due to their lower complexity and, therefore, better linear separability after applying the RFFM feature transform.

5 Conclusion

In this work, we developed a resource-efficient federated HDC method that considerably reduces the computational and communication loads while achieving a comparable or higher predictive performance than that of the baseline federated HDC method. The proposed solution demonstrates up to a $\times 4$ reduction in the computational and communication costs for the selected datasets as compared to the baseline.

ACKNOWLEDGEMENT

This work was supported by the Academy of Finland (projects RADIANT, IDEA-MILL, and SOLID).

REFERENCES

Anguita, D., Ghio, A., Oneto, L., Parra Perez, X., and Reyes Ortiz, J. L. 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*, pp. 437–442, 2013.

Bernstein, J., Wang, Y.-X., Azizzadenesheli, K., and Anandkumar, A. signSGD: Compressed optimisation for non-convex problems. In *International Conference on Machine Learning*, pp. 560–569. PMLR, 2018.

Chandrasekaran, R., Ergun, K., Lee, J., Nanjunda, D., Kang, J., and Rosing, T. FHDnn: Communication efficient and robust federated learning for AIoT networks. In *Proceedings of the 59th ACM/IEEE Design Automation Conference*, pp. 37–42, 2022.

Diao, C., Kleyko, D., Rabaey, J. M., and Olshausen, B. A. Generalized learning vector quantization for classification in randomized neural networks and hyperdimensional computing. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–9. IEEE, 2021.

Dutta, A., Gupta, S., Khaleghi, B., Chandrasekaran, R., Xu, W., and Rosing, T. HDnn-PIM: Efficient in-memory design of hyperdimensional computing with feature extraction. In *Proceedings of the Great Lakes Symposium on VLSI 2022*, pp. 281–286, 2022.

Ge, L. and Parhi, K. K. Classification using hyperdimensional computing: A review. *IEEE Circuits and Systems Magazine*, 20(2):30–47, 2020.

Hernandez-Cane, A., Matsumoto, N., Ping, E., and Imani, M. OnlineHD: Robust, efficient, and single-pass online learning using hyperdimensional system. In *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 56–61. IEEE, 2021.

Hernández-Cano, A., Zhuo, C., Yin, X., and Imani, M. RegHD: Robust and efficient regression in hyperdimensional learning system. In *2021 58th ACM/IEEE Design Automation Conference (DAC)*, pp. 7–12. IEEE, 2021.

Hsieh, C.-Y., Chuang, Y.-C., and Wu, A.-Y. A. FL-HDC: Hyperdimensional computing design for the application of federated learning. In *2021 IEEE 3rd International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, pp. 1–5. IEEE, 2021.

Imani, M., Zou, Z., Bosch, S., Rao, S. A., Salamat, S., Kumar, V., Kim, Y., and Rosing, T. Revisiting hyperdimensional learning for FPGA and low-power architectures. In *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pp. 221–234. IEEE, 2021.

Kairouz, P., McMahan, H. B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A. N., Bonawitz, K., Charles, Z., Cormode, G., Cummings, R., et al. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning*, 14(1–2):1–210, 2021.

Kanerva, P. Hyperdimensional computing: An introduction to computing in distributed representation with high-dimensional random vectors. *Cognitive Computation*, 1: 139–159, 2009.

Kang, J., Khaleghi, B., Kim, Y., and Rosing, T. XCelHD: An efficient GPU-powered hyperdimensional computing with parallelized training. In *2022 27th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 220–225. IEEE, 2022a.

Kang, J., Khaleghi, B., Rosing, T., and Kim, Y. OpenHD: A GPU-powered framework for hyperdimensional computing. *IEEE Transactions on Computers*, 71(11):2753–2765, 2022b.

Karunaratne, G., Le Gallo, M., Cherubini, G., Benini, L., Rahimi, A., and Sebastian, A. In-memory hyperdimensional computing. *Nature Electronics*, 3(6):327–337, 2020.

Kim, M. and Smaragdis, P. Bitwise neural networks. *arXiv preprint arXiv:1601.06071*, 2016.

Kleyko, D., Rachkovskij, D., Osipov, E., and Rahimi, A. A survey on hyperdimensional computing aka vector symbolic architectures, Part I: Models and data transformations. *ACM Computing Surveys*, 55(6):1–40, 2023a.

Kleyko, D., Rachkovskij, D., Osipov, E., and Rahimi, A. A survey on hyperdimensional computing aka vector symbolic architectures, Part II: Applications, cognitive models, and challenges. *ACM Computing Surveys*, 55(9): 1–52, 2023b.

Konečnỳ, J., McMahan, H. B., Ramage, D., and Richtárik, P. Federated optimization: Distributed machine learning for on-device intelligence. *arXiv preprint arXiv:1610.02527*, 2016.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. 2009.

LeCun, Y. The MNIST database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

Nguyen, H. T., Sehwag, V., Hosseinalipour, S., Brinton, C. G., Chiang, M., and Poor, H. V. Fast-convergent federated learning. *IEEE Journal on Selected Areas in Communications*, 39(1):201–218, 2020.

Nunes, I., Heddes, M., Givargis, T., Nicolau, A., and Veidenbaum, A. GraphHD: Efficient graph classification using hyperdimensional computing. In *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1485–1490. IEEE, 2022.

Plate, T. A. Holographic reduced representations. *IEEE Transactions on Neural Networks*, 6(3):623–641, 1995.

Rahimi, A. and Recht, B. Random features for large-scale kernel machines. *Advances in Neural Information Processing Systems*, 20, 2007.

Schlegel, K., Neubert, P., and Protzel, P. HDC-MiniROCKET: Explicit time encoding in time series classification with hyperdimensional computing. In *2022 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8. IEEE, 2022.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

Xiao, H., Rasul, K., and Vollgraf, R. Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

Yan, Z., Wang, S., Tang, K., and Wong, W.-F. Efficient hyperdimensional computing. *arXiv preprint arXiv:2301.10902*, 2023.

Yu, T., Zhang, Y., Zhang, Z., and De Sa, C. M. Understanding hyperdimensional computing for parallel single-pass learning. *Advances in Neural Information Processing Systems*, 35:1157–1169, 2022.

Zhao, Q., Lee, K., Liu, J., Huzaifa, M., Yu, X., and Rosing, T. FedHD: Federated learning with hyperdimensional computing. In *Proceedings of the 28th Annual International Conference on Mobile Computing and Networking*, pp. 791–793, 2022.