RESOURCE-EFFICIENT FEDERATED HYPERDIMENSIONAL COMPUTING

Anonymous Authors[1]

ABSTRACT

In conventional federated hyperdimensional computing (HDC), larger models usually lead to higher predictive performance but also require 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-HDFL) framework alleviates such constraints by training multiple smaller independent HDC sub-models and refining their concatenation using the proposed dropout-inspired procedure. The numerical comparison demonstrates that the proposed framework achieves 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 would 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 core difference between HDC and the conventional ML models is a more hardware-efficient implementation of the training and inference procedures. In HDC, all the data are mapped into 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 the similarity of the bundled data. Based on this property, the inference procedure reduces to computing Hamming or cosine distances between the HD prototypes, which significantly boosts the computational and energy efficiency compared to matrix-to-matrix multiplications employed in ANNs. By the present moment, HDC has been adapted to image (Dutta et al., 2022), time series (Schlegel et al., 2022), graphs (Nunes et al., 2022), text (Kanerva, 2009) classification problems, regression problems (Hernández-Cano et al., 2021), and others. For a detailed discussion on HDC architectures and their applications, we recommend referring to (Kleyko et al., 2023b) and (Kleyko et al., 2023a).

1.2 Related Work

There are several federated HDC implementations that have been recently proposed. The work in (Hsieh et al., 2021) introduced a procedure for federated training of an HDC model. The main idea of the method is to utilize a bipolar HDC encoding to reduce the amount of transmitted information in bits compared to conventional ML models with real-valued parameters. A similar principle was exploited in (Chandrasekaran et al., 2022) with the core difference that the considered HDC model employed 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 standard model averaging can lead to catastrophic performance degradation and, therefore, suggested a modified model update weighing the local and global models to avoid performance drop. A federated decentralized HDC approach for training randomized neural networks was proposed in (Diao et al., 2021), where the HDC model was

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the Machine Learning and Systems (MLSys) Conference Workshop on Resource-Constrained Learning in Wireless Networks. Do not distribute.

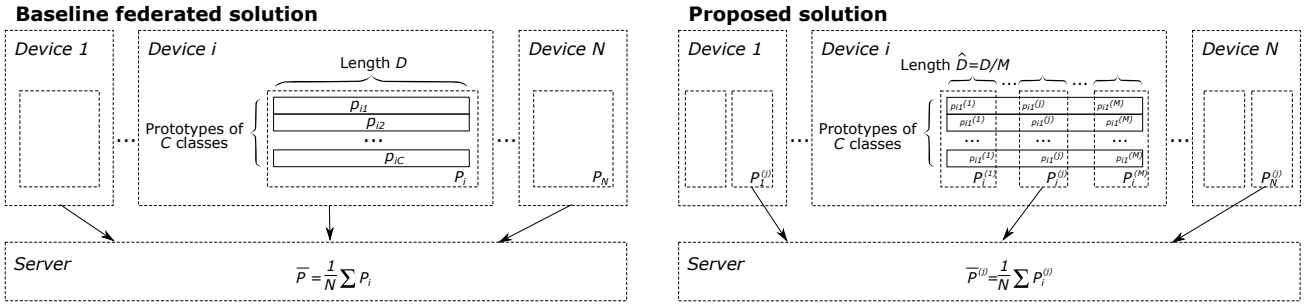

Figure 1. Comparison of proposed RE-HDFL solution and baseline federated HDC.

used as a more computationally-efficient alternative to optimizing the output layer using ordinary least squares. In this method, the properties of HDC were exploited to compress the transmitted HDC model into a single HD vector, thus reducing communication overhead.

Both computational and communication efficiency improvements of the existing implementations of federated HDC are generally achieved via exploiting quantized (binary or bipolar) HD representations. However, we show that we can exploit the properties of HDC to further reduce the computational and communication costs of the federated training. Particularly, we propose a resource-efficient federated HDC method named RE-HDFL, 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 concatenating them into a single one after the federated training. We outline the adopted HDC model and describe the principle of our RE-HDFL below.

2 Federated HDC

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

The training procedure of the HDC model includes three successive procedures: (i) the data transform, (ii) the prototype initialization, and (iii) the prototype retraining. During the data transform procedure, the d-dimensional training data are mapped into 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 assume 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 selecting the 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 the method-specific update rule.

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

2.1 Random Projection-Based Data Transform

The selected HDC mapping should have three essential properties: representation distributiveness, similarity preservation, and implementation efficiency. The first property guarantees that the 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 the similarity of the original d-dimensional data in the HD space, which facilitates robustness of inference over the transformed HD data. The third property implies that the selected mapping θ and similarity measure can be efficiently implemented in hardware.

In our RE-HDFL 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. For the employed random projection-based mapping, we use a cosine distance $\mathrm{dist}(\mathbf{a}, \mathbf{b}) = \frac{\langle \mathbf{a}, \mathbf{b} \rangle}{\|\mathbf{a}\|_2 \|\mathbf{b}\|_2}$ to measure the similarity between two HD vectors $\mathbf{a}$ and $\mathbf{b}$.

The employed random projection-based mapping has sev-

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

	Maximum achieved accuracy						
	RE-HDFL, $D = 5$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 network traffic, i.i.d. setting.

	Rounds to reach baseline $D = 5$K				Total uplink traffic, MB			
	Baseline, $D = 5$K	RE-HDFL, $\hat{D}$			Baseline, $D = 5$K	RE-HDFL, $\hat{D}$		
		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%)**

eral 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, which is parallelized using dedicated libraries such as OpenBLAS, OpenCL, and CUDA, with minimal user intervention. In contrast, the algebraic implementations require processing each data point individually, efficient parallelization of which should 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 system-on-chips (SOCs) with lower energy costs compared to 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 a random projection-based HDC transform a more convenient and universal option for general-purpose mobile platforms, which is the reason we choose it in our federated HDC framework.

2.2 Prototype Construction and Retraining

In our RE-HDFL solution, the prototypes of each of C classes are constructed by summing the HD representations of the corresponding data into a single prototype $\mathbf{p}_i = \sum_{\mathbf{x} \in \mathbf{C}_i} \alpha \cdot \theta(\mathbf{x})$, 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 further retraining procedure is to increase dissimilarity between the prototypes by reinforcing correct predictions and penalizing incorrect ones using the iterative refining procedure for the OnlineHD algorithm proposed in (Hernandez-Cane et al., 2021). If the training data point $\mathbf{x}$ of class i is misclassified as class j, then the corresponding HD prototypes $\mathbf{p}_i$ and $\mathbf{p}_j$ are updated as

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

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 follows the FedAvg algorithm (Konečný 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 the global epoch, the parameter server aggregates the prototypes and returns the following averaged value to the user devices:

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

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.

Table 3. Comparison of maximum achieved accuracy, non-i.i.d. setting.

	RE-HDFL, $D = 5$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

As discussed in (Zhao et al., 2022), the model averaging in (3) can experience catastrophic performance degradation and, therefore, the local HDC models of the participants $\mathbf{P}_j$ should be weighted with the received average $\bar{\mathbf{P}}$. In our experiments, we do not observe any noticeable reduction of catastrophic 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 federated training of HDC models, especially for lower dimensionalities D. We note that a catastrophic performance degradation can be successfully avoided with a proper selection of the federated training hyperparameters.

3 Proposed RE-HDFL Method

In the existing federated HDC solutions, the devices train full-sized D-dimensional HDC models throughout the entire federated training process. While choosing larger values of D generally leads to higher predictive performance, it also results in increased communication overhead 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 transmission of large model updates from multiple (in practice, thousands) devices meets a communication bottleneck, where the time required to receive all the model updates becomes considerably longer than the time needed to compute the model updates (Kairouz et al., 2021). Third, larger sizes D of the HDC model require more available RAM for storing HD representations. Although batched data processing can alleviate this limitation, it may introduce additional computational overhead by repeating the HDC mapping during the retraining procedure.

3.1 Core Idea of Proposed Solution

The proposed RE-HDFL method overcomes the aforementioned limitations by collaboratively training M independent HDC sub-models $\{\mathbf{P}^{(1)}, \dots, \mathbf{P}^{(M)}\}$ of dimensionality $\hat{D} = D/M$ and performing inference on the concatenated HDC model $\mathbf{P}_C = [\mathbf{P}^{(1)}, \dots, \mathbf{P}^{(M)}]$ (see Fig. 1). Our RE-HDFL method consists of two successive stages: federated training and federated refining, having a 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 $\mathbf{P}_C = [\mathbf{P}^{(1)} \dots \mathbf{P}^{(M)}]$ and can perform the inference 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 the selected positions. That is, at each global epoch, the participants update only a subset of positions of the concatenated HDC model $\mathbf{P}$. While the hyperparameters (G_T, G_R) could be customized for a particular task, we observe that it is a best practice to set $G_T = M$, i.e., perform only one global iteration for each HDC sub-model, and allocate the remaining global iteration for federated refining. Even though the refining stage is optional, we observe that it can considerably improve the predictive performance 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-HDFL 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 the transformed original data points. After the original data are projected into 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 re-

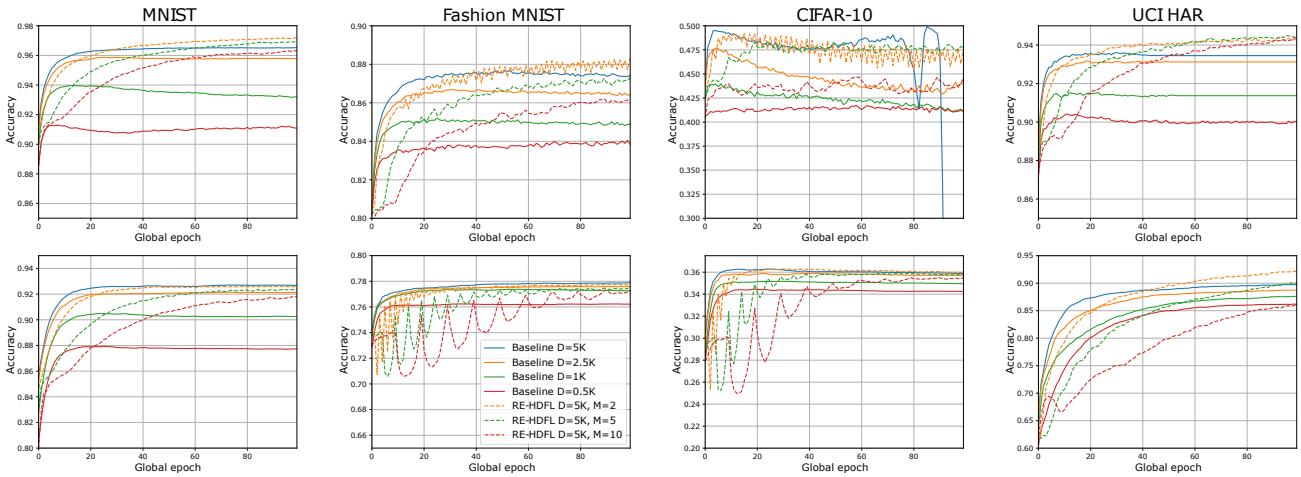

Figure 2. Comparison of performance of RE-HDFL and baseline federated HDC under i.i.d. (top) and non-i.i.d. (bottom) settings.

training procedure involves computing pairwise distances between the HD representations and C HD prototypes with further refining 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-HDFL 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 retraining over a subset of D_0 randomly selected positions of the concatenated HDC model during G_R global epochs. Therefore, the total computational complexity of our RE-HDFL 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 communications 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 by lower training costs, as we demonstrate in Section 4.

3.3 Discussion

Below, we briefly describe the intuition behind the proposed RE-HDFL method leaving a rigorous theoretical analysis for future works. 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 easily 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 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, which 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-HDFL 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 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 randomly dropped to prevent model overfitting. Using a smaller subset of D_0 randomly selected positions to estimate the distance in (4) increases its variance and, 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 the local minimum and continue converging to a better solution.

4 Numerical Results

We compare our RE-HDFL method to the baseline federated HDC implementation. The considered baseline differs from the proposed method in two key points: (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-HDFL solution. This approach is based on the conventional federated learning principles, where the participants upload the entire updated model, and has been adapted 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-HDFL solution to the baseline federated HDC using 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. settings. In the i.i.d. scenario, the participants have identical or close-to-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), 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-HDFL method is compatible with arbitrary feature extractors, including state-of-the-art ANN-based solutions (Dutta et al., 2022), and, hence, its feature extraction stage is entirely implementation-specific. We set the number of RFFM features to 3.2K and the RFFM length-scale parameters to $\sigma = 1$ for the image datasets and $\sigma = 2.5$ 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 $D = 5\text{K}$, $D_0 = \hat{D}$, and vary the number of the 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 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 non-i.i.d. scenarios, respectively.

In Table 1, we provide the maximum accuracy achieved by the considered federated HDC methods under the i.i.d. scenario after G global iterations. One may see that the proposed RE-HDFL method achieves comparable or higher accuracy than that of the baseline while processing smaller HDC models with lower 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 $D = 5\text{K}$ as well as the corresponding total volume of data transmitted by the participants. The results demonstrate that our RE-HDFL method can reach the network traffic reduction of up to $\times 4$ times without sacrificing the predictive performance. The comparison of the maximum accuracy of the methods under the non-i.i.d. scenario is provided in Table 3.

In Fig. 2, we compare the accuracy evolution of our RE-HDFL and the baseline federated HDC methods. We hypothesize that the observed instability under the i.i.d scenario is mainly due to the imperfect selection of a feature extractor. The works in (Dutta et al., 2022; Chandrasekaran et al., 2022) demonstrate a significantly smoother convergence with a CNN-based feature extractor, and, therefore, our RE-HDFL method could be further enhanced by applying more advanced data preprocessing techniques. For 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 the stability of the federated training. On 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 using the *concatenated* HDC model, and by the M-th iteration, all D positions of the HDC model become updated, resulting in the accuracy leap. We do not observe such behavior on MNIST and UCI HAR datasets 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 computational and communication load while achieving comparable or higher predictive performance. The proposed solution demonstrates up to a $\times 4$ reduction in computational and communication costs on the selected datasets compared to the baseline federated HDC solution.

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 II: Applications, cognitive models, and challenges. *ACM Computing Surveys*, 55(9): 1–52, 2023a.

Kleyko, D., Rachkovskij, D. A., Osipov, E., and Rahimi, A. A survey on hyperdimensional computing aka vector symbolic architectures, part I: Models and data transformations. 55(6):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. Understanding hyperdimensional computing for parallel single-pass learning. *arXiv preprint arXiv:2202.04805*, 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.