HYPERFEEL: AN EFFICIENT FEDERATED LEARNING FRAMEWORK USING HYPERDIMENSIONAL COMPUTING

Anonymous authors
Paper under double-blind review

ABSTRACT

Federated Learning (FL) aims to establish a shared model across decentralized clients under the privacy-preserving constraint. Each client learns an independent model with local data, and only model updates are communicated. However, as the FL model typically employs computation-intensive neural networks, major issues in Federated Learning are (i) significant computation overhead for local training; (ii) the massive communication overhead that arises from sending around the model updates; (iii) notable performance degradation resulting from the non-IID scenario.

In this work, we propose HyperFeel, an efficient learning framework for federated learning based on Hyper-Dimensional Computing (HDC), that can significantly improve communication/storage efficiency over existing works with nearly no performance degradation. Unlike current solutions that employ neural networks as the learned model, HyperFeel introduces a simple yet effective computing paradigm that encodes and represents data using hyperdimensional vectors. Then, it performs concise and highly parallel operations for encryption, computation, and communication, taking advantage of the lightweight feature representation of hyperdimensional vectors. For further enhance HyperFeel performance, we propose a two-fold optimization scheme combining the characteristics of encoding and updating in hyper-dimensional computing. On the one hand, we design the personalization update based on hyperdimensional computing with a client-specific model, which achieves better accuracy to the non-IID data. On the other hand, we extend the framework from horizontal FL to vertical FL based on a shared encoding mechanism. Comprehensive experimental results demonstrate our method consistently outperforms the state-of-the-art FL models. Typically, we achieves $26\times$ storage reduction and up to $81\times$ communication reduction over FeAVG, with minimal accuracy drops on FEMNIST and Synthetic. *Code will be open-source in the camera-ready version.*

1 INTRODUCTION

With the advent of the Internet of Things (IoT), the amount of at-the-edge intelligent devices has grown dramatically over the past years, generating massive volumes of data daily Bonawitz et al. (2019).Meanwhile, deep learning models on these devices for personalized, low-latency AI applications. Therefore, there is significant potential to leverage the data provided by edge devices to learn deep learning models locally, as edge devices equipped with increased computing power are becoming smarter, together with attention to data privacy Hitaj et al. (2017). This has draw growing research interests in Federated Learning (FL), which allows multiple participants to jointly train a deep learning model on their integrated data without requiring any participant to disclose their local data to a centralized server Wang et al. (2020).

The key insight behind FL protocols is the central model in the server trained using data stored locally on the clients Li et al. (2020). In each round of the training procedure, models stored in clients are trained with local data and then collaboratively trained by uploading model parameters or model gradients to the server, and finally the server sends the central model to clients. These steps are repeated until the convergence criterion is satisfied Sattler et al. (2019).

Although FL serves as an effective method for privacy-preserving learning, model train brings significant overhead. The overhead lies in two aspects: (i) massive communication overhead that arises from sending around the model updates McMahan et al. (2017); Sattler et al. (2019); (ii) massive computation overhead due to performing neural network training Wang et al. (2020). However, modern network model may require up to GBs (Giga Bytes) for model size and 10^2 GFLOPs (Giga Floating Point Operations) for a single forward propagation Brown et al. (2020). Over the hundreds or thousands of training iterations on large datasets, for each client, the total communication can easily grow to more than one TeraByte and the total computation up to TeraFLOPS Li et al. (2020). Besides, the local data for each client can be non-IID (non-independent and identical distribution) Sattler et al. (2019); Li et al. (2019b), which will degrade the model performance on a specific device download from the centralized server after training. These make FL far from practical for real-world deployment.

To address the aforementioned challenges, we explore an energy-efficient FL regime. We optimize the FL by redesigning the update process based on a new computing paradigm, Hyper-Dimensional Computing (HDC) Zou et al. (2022). HDC is a promising alternative of conventional machine learning models. The key idea of HDC is to represent data using high-dimensional vectors (called hypervectors), which allows energy-efficient vector operations based on its massively parallel computation flow Imani et al. (2021). The HDC provides several features that make it well-suited to FL: (i) it transforms complex computation of feature learning processes to similarity matching, (ii) HDC can naturally enable secure and light-weight learning, and (iii) it provides strong robustness to noise – a key strength for IoT scenarios. These features make HDC a promising solution for today's edge devices with limited resources.

In this paper, we propose HyperFeel, a Hyper-Dimensional federated learning framework. Instead of working with raw data, HyperFeel maps the raw data into high-dimensional space and performs vector operations with simple and parallel similarity matches. The main contributions of the paper are summarized as follows:

- To the best of our knowledge, HyperFeel is the first end-to-end learning framework for both horizontal and vertical federated learning based on HyperDimensional computing. HyperFeel revisits the feature learning with brain-like memorization that HDC natively supports. HyperFeel creates a reference library that stores category vectors for each client and the centralized server by memorizing the patterns in high-dimension, enabling the main vector operations to process in a hardware-friendly way.
- We design the online update mechanism with single sample training implemented in local clients, empowering HyperFeel with stronger adaptability. This includes supporting similarity matching as well as local aggregation followed by uploading to the server.
- We propose an optimization scheme for scalability issues in the FL due to the non-IID data. HyperFeel allows clients to selectively update the hypervectors stored locally when downloading models with low communication overhead, allowing them to maintain their personalized reference library.
- We evaluate HyperFeel efficiency on a wide range of dataset. On average, our solution provides $26\times$ and $81\times$ storage efficiency improvement and communication reduction compared to the state-of-the-art FL algorithms with almost no accuracy loss.

2 Preliminaries and Motivation

2.1 Federated Learning

FL deals with learning a central model (i.e. the server) in privacy-constrained scenarios, where data are stored on multiple devices (i.e. the clients) Li et al. (2020). A major problem in federated learning is the enormous communication overhead incurred by sending model updates. In the field of communication efficient distributed deep learning, various methods have been proposed to reduce the amount of communication during training, including: decreasing the frequency of communication McMahan et al. (2017); Strom (2015), reducing the amount of communication (i.e., sparsification on transmitted data) Konečnỳ et al. (2016); Aji & Heafield (2017), quantization of communication data (i.e., restricting updates to a reduced set of values) Bernstein et al. (2018). However, these methods in federated learning are inefficient because the communication from the

server to the client is not considered. FedAvg McMahan et al. (2017) is a simple but effective baseline for FL that takes into account both the upstream and downstream communication, but it is a neural network-based approach that leads to communication and computation overhead remaining high level.

2.2 HYPER-DIMENSIONAL COMPUTING (HDC)

HDC maps all data points into high-dimensional space to mimic the behaviors of the brain computed with patterns of neural activity, allowing energy-efficient computation based on its massively parallel computation flow Rahimi et al. (2016); Zou et al. (2022). HDC is well suited to tackle learning tasks for edge devices with limited resources and computing power Imani et al. (2021); Liu et al. (2022). The key insight of HDC is to represent the information with a hypervector, which is the fundamental "block" of HDC, and the similarity between information that contained in two hypervectors. The dimensionality should be large enough (usually higher than 10,000 dimensions) to ensure two high dimensional vectors that are initialized randomly from the polarization values (i.e., {-1, 1} or {0,1}) are approximately orthogonal Ma & Jiao (2022). An hypervector with D dimensions can be denoted as $H = \{h_1, h_2, \cdots, h_D\}$. The encoding keeps the main information of raw data as a pattern of values in high-dimensional space. Further, HDC encodes the data to extract features using well-defined efficient basic operations: (i) Bundling $+$ is an addition operation of multiple hypervectors into a single hypervector, (ii) Binding $\times$ performs element-wise multiplication of two hypervectors with the same dimension, and (iii) Permutation ρ performs rotational shift over a single hypervector.

2.3 MOTIVATION

In federated learning, the distribution of training data and computational resources is a fundamental and fixed property of the learning environment. This imposes the following challenges.

Non-IID data: In real-world FL tasks, data distributions of different clients may vary since the data are usually collected from different sources or scenarios. Existing FL approaches often simply ignore the distribution discrepancy, leading to notable performance drops under the the non-IID scenarios Caldarola et al. (2021).

Massive Communication: Communication is a key bottleneck for FL because the data generated on each device must remain local to ensure data privacy, the client inevitably communicates with the centralized server at each model update in the training process. Indeed, federated networks potentially comprise a massive number of devices, leading to direct communication for weight updates becoming infeasible since the communication cost is growing linearly along with the number of clients and training rounds.

Limited Resource: The resource-constrained edge devices are typically not directly charged, making their computing power constrained by the battery with finite capacity. For neural networks, it is significantly expensive to perform iterations of backward-propagated gradients. Besides, edge devices also typically have only very limited memory. As neural networks are becoming deeper and wider to achieve better performance, it is challenging to deploy them on edge devices.

Therefore, we need alternative learning methods for federated learning that can train on the less-powerful IoT devices, to eliminate the mismatch of hardware constraints and the neural network models, so that the high-cost communication and computation can be alleviated without causing severe accuracy drops.

3 APPROACH

3.1 OVERALL FRAMEWORK

Figure 1 shows an overview of HyperFeel federated learning in the highdimensional space. In HDC, all training and inference computations are performed on the encoded data in the high-dimensional space. The first step of HyperFeel is to map the raw data (i.e., image) into a high-dimensional space. During encoding, the input samples are "encoded" into their representative hypervectors using a set of basic operations and the item memories. HyperFeel aggregates all encoded hypervectors representing features to generate a reference class locally stored in clients, called the HDC library (i.e., Associate Memory, AM). The HDC library consists of k hypervectors (where k is the number of classes), where each hypervector with D dimensions stores thousands of features belonging to the same class. Considering the online training, we adopt a strategy of batch uploading updates to

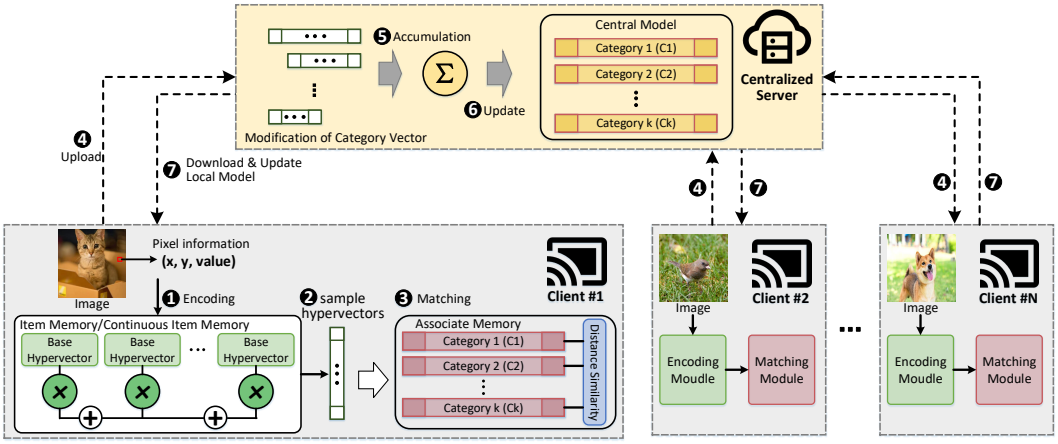

Figure 1: Overall Flow of HyperFeel Framework. First, we map the raw data into a high-dimensional space ①. Then, HyperFeel combines all encoded features to generate the sample hypervector ②. Next, HyperFeel checks the similarity of the encoded sample with all reference hypervectors stored in the memory ③. For each mis-predicted sample, clients transfer modifications on the incorrectly predicted class vector to the server ④. After collecting ⑤ the modification on the class vectors, the server updates the central model ⑥ and sends it back to each client for the local update ⑦.

construct the central model on the centralized server and download it by clients as the initial local model at the end of training. After that, each client will conduct several rounds of retraining based on the local data. During this period, we design an adaptive schedule for the learning rate based on the local sample distribution and adjust the local model with the aid of the globally aggregated update hypervector to achieve fast convergence.

3.2 HyperFeel Encoding

Instead of representing the features using their value, HDC represents them using a set of hypervectors. Encoding is the basic phase of HDC model. During encoding, the input samples are transformed into the representation in the highdimensional space through the Item Memory (IM) or Continuous Item Memory (CIM). Then, the information is encoded into a hypervector with higher-order meaning using the basis operations of hyperdimensional computing.

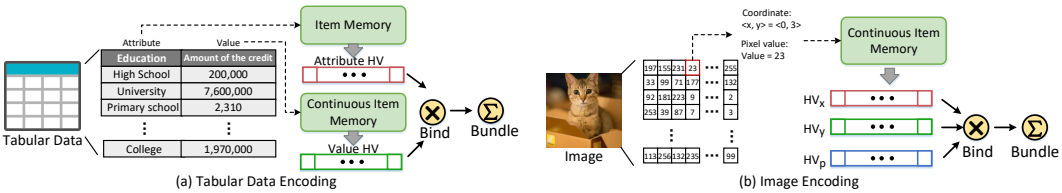

Figure 2: Encoding the tabular data (a) and image (b) into high-dimensional space in HyperFeel.

(i) Tabular Data Encoding: As shown in Figure 2(a), in the tabular data (bank data or synthetic dataset) encoding, each attribute and its value are mapped by IM and CIM to obtain the corresponding hypervector, respectively. Then, both of them are integrated together using the binding operation. Next, all attribute key-value pairs of a sample are incorporated into a sample hypervector $\vec{Q}$ using the bundling operation.

(ii) Image Encoding: As shown in Figure 2(b), in the image encoding, we feed the pixel coordinates $< x, y >$ and pixel value p into CIM for mapping the values into the hyperdimensional space to obtain the hypervectors HV_x, HV_y, HV_p, respectively, and then combine the three into a representation vector of pixel points by the binding operation. Next, we utilize the bundling operation to accumulate the representation vectors of all pixels in the image to form the hypervector of the sample.

Suppose the input sample has M pixels $\vec{F} = \{< f_x^1, f_y^1, f_p^1 >, < f_x^2, f_y^2, f_p^2 >, \cdots, < f_x^M, f_y^M, f_p^M >\}$, our encoding represents this sample as:

$$\vec{Q} = \sum_{i=1}^{M} HV_x^i HV_y^i HV_p^i \tag{1}$$

where, $\vec{Q}$ is the (non-binary) aggregation and $HV^i_{x,y,p}$ is the (binary) hypervector corresponding to i-th pixel information $< x, y, p >$ of the sample, obtained from the CIM by using the information of the i-th pixel as indices (i.e., $HV^i_{x,y,p} = CIM(f^i_{x,y,p})$).

3.3 HYPERFEEL SIMILARITY MATCHING

As a lightweight classifier, the operations in the HDC need to be hardware friendly, where the similarity matching can be performed efficiently under limited resources. The similarity matching operation depends on the distance metric, which depends on the data representation of the elements in the hypervector. For hypervectors with a binary representation, HyperFeel only needs to employ the Hamming distance as the similarity metric, which is a fast and efficient way to return matching results. Still, the binary representation allows only a limited amount of information to be stored for each element in the hypervector, resulting in compromised model accuracy. On the other hand, HyperFeel uses a higher precision (e.g., 32-bit values) representation of the hypervectors, which significantly increases the information representation capability of the hypervectors and allows more information to be stored in each hypervector. However, HyperFeel needs to adopt the cosine distance as a similarity metric within such a high-precision representation.

$$cosine\ similarity = \frac{a \cdot b}{\|a\|\|b\|} \tag{2}$$

We perform matching of the sample hypervector $\vec{Q}$ with all the class hypervectors in AM $C = \{C_1, C_2, \cdots, C_k\}^T$ and return the class with the highest similarity. Therefore, we only need to determine the relative magnitude of the vector Q and the class vectors C_i in AM. From Eq.(3), the sample hypervector norm $\|\vec{Q}\|$ is the same for all the class hypervectors in AM during the similarity matching. Coupled with the fact that the class hypervectors in the model can be determined after training, we can simplify the cosine similarity to the inner product of the sample hypervectors and the class hypervectors.

$$\begin{aligned} &C' = [C_1/\|C_1\|, C_2/\|C_2\|, \cdots, C_j/\|C_k\|]^T \\ &\vec{Q}' = \vec{Q}/\|\vec{Q}\| \\ &cosine\ similarity\ vector = C' * \vec{Q}' = [\frac{Q^T C_1}{\|\vec{Q}\|\|C_1\|}, \frac{Q^T C_2}{\|\vec{Q}\|\|C_2\|}, \cdots, \frac{Q^T C_k}{\|\vec{Q}\|\|C_k\|}]^T \end{aligned} \tag{3}$$

3.4 OFFLINE AND ONLINE TRAINING

In HDC, training is performed by element-wise addition of all encoded hypervectors samples in each existing class. HyperFeel maintains k local class hypervectors (initially all-0 vectors) in each client, where k is the number of classes. Once a new sample data is generated locally by the client, online training is activated. The sample is encoded and bundled to the corresponding class hypervector. Once the amount of newly collected samples reaches a pre-defined threshold, the class vectors will be uploaded to the centralized server, and the locally stored vectors will be restored. There is a central AM stored on the centralized server for accumulating the class hypervectors from all clients, which is downloaded to each client as the initial local model after the online training.

To further optimize the model performance, HyperFeel introduces a retrain strategy applicable to federated learning, where several rounds of federated retraining are performed based on the initial local model. In one-round federated retraining, the samples from the client are first encoded one by one. Then, we exploit the encoded vectors to match the similarity with the class hypervectors in the local AM, and the prediction class is obtained. Next, we check whether the prediction class is correct. If no, we update the local model and accumulate the modified vectors according to the classes with additional storage.

$$\begin{aligned} C_{ix} &= C_{ix} + lr \cdot Q \\ C_{iy} &= C_{iy} - lr \cdot Q \\ \delta_{ix} &= \delta_{ix} + lr \cdot Q \\ \delta_{iy} &= \delta_{iy} - lr \cdot Q \end{aligned} \tag{4}$$

where, C_{ix} is the vector of the true class, C_{iy} is the vector of the predicted class, δ_{ix} is the modified vector of the true class, δ_{iy} is the modified vector of the predicted class, lr is the learning rate and

Algorithm 1: Retraining Strategy for the HyperFeel

Data: number of retrain rounds R, number of clients N, number of classes K
Result: models $C_1, \ldots, C_N$

for $r \leftarrow 1$ *to* R **do**
  for $i \leftarrow 1$ *to* N ***in parallel*** **do**
    Client$_i$ does:
    $\Delta \leftarrow$download$_{Server \rightarrow Client_i}(\Delta)$;
    for $j \leftarrow 1$ *to* K **do**
      $C_{ij} \leftarrow$Personalization Update(C_{ij}, Δ_j);
    end
    for $sample, label_{real} \in local\ data$ **do**
      $Q \leftarrow encode(sample)$;
      $label_{predict} \leftarrow AssociativeSearch(Q, C_i)$;
      if $label_{predict} \neq label_{real}$ **then**
        $C_i \leftarrow$Accumulation(C_i, Q, $label_{real}$, $label_{predict}$);
        $\delta_i \leftarrow$Accumulation(δ_i, Q, $label_{real}$, $label_{predict}$);
      end
    end
    upload$_{Client_i \rightarrow Server}(\delta_i)$;
  end
  Server does:
  gather$_{Client_i \rightarrow Server}(\delta_i),\ i = 1, 2, \ldots, N$;
  for $j \leftarrow 1$ *to* K **do**
    $\Delta_j \leftarrow \sum_{i=1}^{N} \delta_{ij}$;
  end
  broadcast$_{Server \rightarrow Client_i}(\Delta),\ i = 1, 2, \ldots, N$;
end

the sample vector is $\vec{Q}$. After one-round retraining, the client uploads the locally modified vectors to the server for aggregation to obtain the global modified vector $\{\Delta_1, \Delta_2, \ldots, \Delta_K\}$ and then clients download it for adjusting the local model again. We introduce a personalization strategy based on local data distribution during the adjusting process.

$$\Delta_j = \sum_{i=1}^{N} \delta_{ij} C_{ij} = C_{ij} + \frac{error_{ij}}{cnt_{ij}} lr \cdot \Delta_j \tag{5}$$

where, cnt_{ij} and $error_{ij}$ are the total number of samples of class j on the client i and the number of samples with errors in one-round retraining, respectively. The retraining strategy is formalized in Algorithm 1.

3.5 Extension to Vertical Federated Learning

In the horizontal federated learning, we upload the modifications to the hypervectors of mispredicted data. Since the HDC model performs encoding by projecting and encoding the raw data to hypervectors, this unique computing paradigm of HDC ensures the encryption of the client and server transmission and achieves the purpose of privacy-preserving.

In the vertical federated learning, based on the assumption that "the server is honest and does not collude with clients, but all the clients are honest but curious to each other." Yang et al. (2019), the participating clients build local IMs to store the attribute hypervectors separately, and all the clients share a CIM to store the hypervectors of attribute values. Here, the participating clients adopt the identical PSI (Privacy Set Intersection) approach as Cristofaro et al. (2010) for data alignment, and then the clients encode the raw data and upload them consistently. The client with the tag transfer tag in public-private key encrypted way with the server, and then the server accumulates the encoded data as well as the class vector updates. During inference, the encoded data of each client is uploaded. Then the server accumulates them to get the target hypervectors and feeds them into the AM located at the server to match and return the inference result to clients.

3.6 OVERHEAD ANALYSIS

We make detailed discussion on the overhead. Table 1 reports the storage cost and single-round communication cost for the baseline models FedAVG and HyperFeel. N, M and C represent the number of clients, the amount of model parameters, and the fraction of clients that perform computation on each round. In proposed HyperFeel, the storage and communication costs per client are determined by the HDC memories, i.e., the dimension D of HV and the number of classes K. HyperFeel has a significant theoretical advantage over the NN-based FedAVG, both in terms of storage and communication costs.

Table 1: The comparison of storage and communication cost between FedAVG and HyperFeel theoretically.

Method	Storage Cost	Comm. Cost(per round)
FedAvg	N×M	C×N×M
HyperFeel	D×N×K	D×N×K

4 RESULTS

Table 2: The comparison of validation accuracy, storage and communication cost between HyperFeel and benchmarks in the horizontal federated learning.

Method	**Storage Cost**	**Communication Cost**	**Accuracy**
FEMNIST DataSet			
FedAvg McMahan et al. (2017)	6.35MB	5.58GB	74.72%
Additional pipeline Caldas et al. (2018)	-	-	80.24%
FL-HDC Hsieh et al. (2021)	2.37MB	2.08GB	48.99%
HyperFeel (ours)	249.63KB	70.95MB	68.54%
Synthetic DataSet			
FedAvg McMahan et al. (2017)	1.19KB	1.17MB	71.89%
Additional pipeline Caldas et al. (2018)	-	-	87.34%
FL-HDC Hsieh et al. (2021)	195.31KB	183.11MB	79.69%
HyperFeel (ours)	27.47KB	6.10MB	90.13%

4.1 EXPERIMENT SETUP

We use the datasets FEMNIST LeCun (1998); Afshar et al. (2017) and Syhthetic Li et al. (2019a), which are specifically designed for the horizontal federated learning, and the credit card customer default probability prediction dataset Yeh & Lien (2009), which is applicable to the vertical federated learning. For the non-IID setting under the federated learning, whose data of each category are unevenly distributed among clients, we adopt the identical data partition strategy as McMahan et al. (2017). We utilize the federated learning benchmark framework LEAF Caldas et al. (2018) to process and partition the FEMNIST and Synthetic datasets, while for the credit card customer default probability prediction dataset, we employ the full set of them for evaluating our extended vertical federated learning model. We evaluate our method using several representative benchmark, including FedAvg McMahan et al. (2017), FedAvg with additional pipeline Caldas et al. (2018), FL-HDC Hsieh et al. (2021), used in horizontal federated learning, and K-Nearest Neighbor (KNN) Math et al. (2021), Logistic Regression (LR) Yeh & Lien (2009), Parsimonious Bayesian Odiathevar et al. (2022), and Neural Network (NN) Romanini et al. (2021) used in vertical federated learning. For the hyperparameter setting, we set $D = 1000$ in HyperFeel, where the performance of HyperFeel saturates. To fairly compare with the beaseline methods, we set the number of clients $k = 30$. In this work, all experiments are performed on a 2.10GHz Intel(R) Xeon(R) Silver 4208 CPU with 128GB DRAM. under the framework of Pytorch Geometric Paszke et al. (2019).

4.2 PERFORMANCE RESULTS

We first compare the accuracy of our proposed method with the baseline methods. Table 2 summarizes the results of the present method and previous methods on the FEMNIST dataset and the Synthetic dataset in horizontal federated learning. Compared with FedAvg, a widely used neural network-based federal learning method, HyperFeel achieves a close performance on relatively

complex image classification tasks, and a higher accuracy on the Syhthetic dataset. Moreover, our HyperFeel is extremely competitive in terms of model storage, computation and communication overhead. For example, regarding the Synthetic dataset, our HyperFeel algorithm shows better accuracy (90.13%) compared to traditional FL methods FedAvg (71.89%) and Additional pipeline (87.34%) and 10.4% accuracy improvement over the naive combined FL and HDC. Taking FEMNIST dataset as an example, compared to FedAvg, HyperFeel achieves nearly $26\times$ storage reduction and $80.5\times$ communication reduction, because HyperFeel employs lightweight HDCs as backbone instead of traditional neural networks. During the training, the transmission between the client and the server is only a modification of the vector, not the complete one.

Table 3: The comparison of validation accuracy, storage and communication cost between HyperFeel and benchmarks in the vertical federated learning.

Method	**Accuracy**	**Storage**	**Communication**
KNN	84%	182,169KB	182,169KB
LR	82%	-	-
Bayesian	79%	813KB	813KB
Neural Network	83%	2,412KB	2,412KB
HyperFeel (ours)	82%	430.91KB	39.06KB

Table 3 shows our approach shows significantly increased efficiency with almost no accuracy loss, achieving average $113\times$ improvement in the storage cost and more than $1257\times$ improvement in the communication cost for the vertical federated learning.

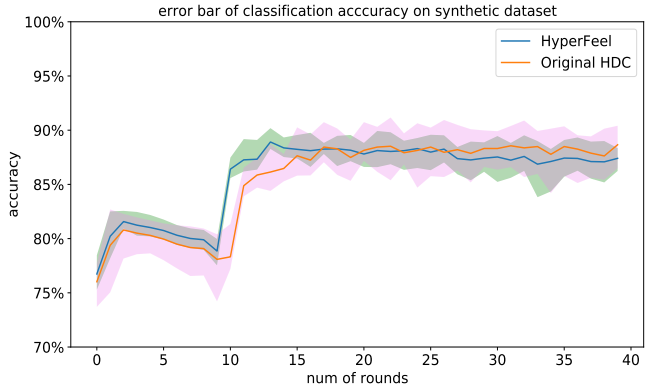

Figure 3: The accuracy evolution curve versus the number of rounds under HyperFeel on synthetic data. We run 5 experiments and the region in shadow indicate the error-band. The original HDC represents training the model directly on the client without online learning.

We first evaluate the impact of the proposed online update mechanism on HyperFeel. As Figure 3 shows, HyperFeel converges faster compared to the one without online training. Note that we use the same number of training rounds. One is offline training locally on the client and one is online training under the federated learning scenario. From the plot, we can find that HyperFeel fits well with the federated learning online scenario and can reach convergence faster, thus further reducing the communication and computational overhead. This corroborates that the present method is a successful migration of HDC in the field of federated learning.

We further investigate the performance of HyperFeel under the non-IID data scenario. We collect the inference accuracies for all clients and report them again in Figure 4. FL-HDC directly binarizes the transmitted vectors to reduce the communication overhead in federated learning, and this direct approach leads to a decrease in accuracy. On the other hand, HyperFeel achieves a better result over the FL-HDC even without introducing the personalization strategy. This is because our scheme is designed from the beginning with the communication overhead in mind, so only information about the modification of the vector is transmitted instead of the complete vector during the transmission. In addition, HyperFeel, combined with the personalization update strategy, further improves the accuracy across clients. This is because the personalization update strategy makes each client maintain

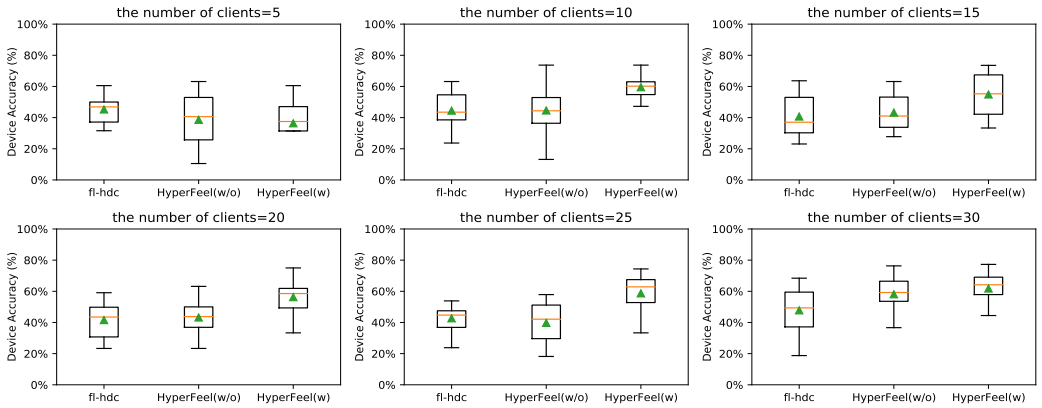

Figure 4: Comparison of FL-HDC and HyperFeel with (w) or without (w/o) personalization update strategy.

the local HDC memories, which can be regarded as a mapping relationship between the raw data and the hypervector, so that each client can encode the data adaptively.

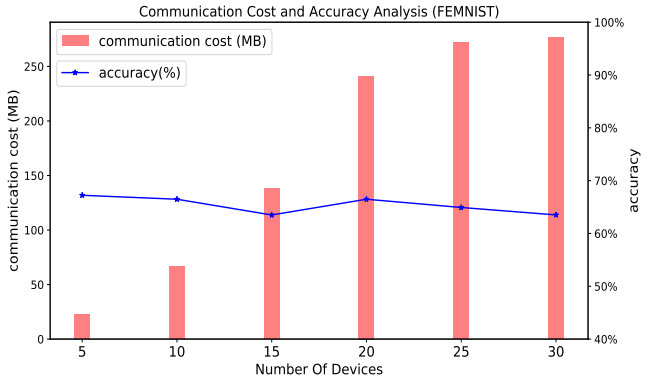

Figure 5: Communication costs and accuracy of proposed HyperFeel with the various number of clients.

Since the number of clients affects the accuracy and the communication cost of the federated learning method, we want to determine the appropriate number of clients by balancing the relationship between the model accuracy (higher is better) and the communication cost (lower is better). Figure 5 shows the impact of the number of clients for the FeMNIST dataset. We sweep it from 5 to 30. Generally, a higher number of clients means more sources to collect data and allows for more information stored in HDC memories, but it will increase the communication cost. From this plot, however, we can also find that it is not consistent in federated learning, because as the number of clients grows, the collected data varies increasingly, introducing noise for the global model and thus slightly degrading the inference accuracy of clients. In summary, the inference accuracy of HyperFeel is barely affected by varying the number of clients, while the total communication overhead increases as the number of clients rise. Moreover, the average communication overhead of a single client is only about $8MB$, and the storage overhead accounts for only about $240KB$, demonstrating that our solution shows great resource-saving advantages in terms of storage and communication.

5 Conclusion

This work proposes an energy-efficient learning framework for federated learning, namely HyperFeel. Thanks to the lightweight representation and efficient computing paradigm of HDC, this method can effectively decrease the storage and communication overhead of the model. Meanwhile, we propose online training and model adaptation strategies, allowing our method to not only further optimize the training accuracy, but also to extend from horizontal federated learning to support vertical federated learning. HyperFeel achieves more than $80\times$ improvement in communication efficiency while maintaining competitive performance, as validated in the comprehensive experiments across various benchmarks.

REFERENCES

Saeed Afshar, J Tapson, Andre van Schaik, et al. Emnist: An extension of mnist to handwritten letters. *Retrieved September*, 17:2018, 2017.

Alham Fikri Aji and Kenneth Heafield. Sparse communication for distributed gradient descent. *arXiv preprint arXiv:1704.05021*, 2017.

Jeremy Bernstein, Yu-Xiang Wang, Kamyar Azizzadenesheli, and Animashree Anandkumar. signsgd: Compressed optimisation for non-convex problems. In *International Conference on Machine Learning*, pp. 560–569. PMLR, 2018.

Keith Bonawitz et al. Towards federated learning at scale: System design. *Proceedings of Machine Learning and Systems*, 1:374–388, 2019.

Tom Brown et al. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 1877–1901. Curran Associates, Inc., 2020.

Debora Caldarola et al. Cluster-driven graph federated learning over multiple domains. In *ICCV*, pp. 2749–2758, 2021.

Sebastian Caldas, Sai Meher Karthik Duddu, Peter Wu, Tian Li, Jakub Konečnỳ, H Brendan McMahan, Virginia Smith, and Ameet Talwalkar. Leaf: A benchmark for federated settings. *arXiv preprint arXiv:1812.01097*, 2018.

Emiliano De Cristofaro, Jihye Kim, and Gene Tsudik. Linear-complexity private set intersection protocols secure in malicious model. In *International Conference on the Theory and Application of Cryptology and Information Security*, pp. 213–231. Springer, 2010.

Briland Hitaj et al. Deep models under the gan: information leakage from collaborative deep learning. In *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*, pp. 603–618, 2017.

Cheng-Yen Hsieh, Yu-Chuan Chuang, and An-Yeu Andy Wu. 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.

Mohsen Imani et al. 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.

Jakub Konečnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

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

Tian Li, Maziar Sanjabi, Ahmad Beirami, and Virginia Smith. Fair resource allocation in federated learning. *arXiv preprint arXiv:1905.10497*, 2019a.

Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine*, 37(3):50–60, 2020.

Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. On the convergence of fedavg on non-iid data. *arXiv preprint arXiv:1907.02189*, 2019b.

Fangxin Liu, Haomin Li, Xiaokang Yang, and Li Jiang. L3e-hd: A framework enabling efficient ensemble in high-dimensional space for language tasks. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 1844–1848, 2022.

Dongning Ma and Xun Jiao. Hyperdimensional computing vs. neural networks: Comparing architecture and learning process. *arXiv preprint arXiv:2207.12932*, 2022.

Sa Math, Prohim Tam, and Seokhoon Kim. Reliable federated learning systems based on intelligent resource sharing scheme for big data internet of things. *IEEE Access*, 9:108091–108100, 2021.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pp. 1273–1282. PMLR, 2017.

Murugaraj Odiathevar, Winston KG Seah, and Marcus Frean. A bayesian approach to distributed anomaly detection in edge ai networks. *IEEE Transactions on Parallel and Distributed Systems*, 2022.

Adam Paszke, Sam Gross, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Abbas Rahimi et al. Hyperdimensional biosignal processing: A case study for emg-based hand gesture recognition. In *2016 IEEE International Conference on Rebooting Computing (ICRC)*, pp. 1–8. IEEE, 2016.

Daniele Romanini et al. Pyvertical: A vertical federated learning framework for multi-headed splitnn. *arXiv preprint arXiv:2104.00489*, 2021.

Felix Sattler, Simon Wiedemann, Klaus-Robert Müller, and Wojciech Samek. Robust and communication-efficient federated learning from non-iid data. *IEEE transactions on neural networks and learning systems*, 31(9):3400–3413, 2019.

Nikko Strom. Scalable distributed dnn training using commodity gpu cloud computing. In *Sixteenth annual conference of the international speech communication association*, 2015.

Hongyi Wang, Mikhail Yurochkin, Yuekai Sun, Dimitris Papailiopoulos, and Yasaman Khazaeni. Federated learning with matched averaging. *arXiv preprint arXiv:2002.06440*, 2020.

Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2019.

I-Cheng Yeh and Che-hui Lien. The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients. *Expert systems with applications*, 36(2): 2473–2480, 2009.

Zhuowen Zou et al. Biohd: an efficient genome sequence search platform using hyperdimensional memorization. In *Proceedings of the 49th Annual International Symposium on Computer Architecture*, pp. 656–669, 2022.