BHerd: Accelerating Federated Learning by Selecting Beneficial Herd of Local Gradients

Ping Luo, Xiaoge Deng, Ziqing Wen, Tao Sun, Dongsheng Li

***Abstract*—In the domain of computer architecture, Federated Learning (FL) is a paradigm of distributed machine learning in edge systems. However, the systems' Non-Independent and Identically Distributed (Non-IID) data negatively affect the convergence efficiency of the global model, since only a subset of these data samples is beneficial for accelerating model convergence. In pursuit of this subset, a reliable approach involves determining a measure of validity to rank the samples within the dataset. In this paper, we propose the BHerd strategy, which selects a beneficial herd of local gradients to accelerate the convergence of the FL model. Specifically, we map the distribution of the local dataset to the local gradients and use the Herding strategy to obtain a permutation of the set of gradients, where the more advanced gradients in the permutation are closer to the average of the set of gradients. These top portions of the gradients will be selected and sent to the server for global aggregation. We conduct experiments on different datasets, models, and scenarios by building a prototype system, and experimental results demonstrate that our BHerd strategy is effective in selecting beneficial local gradients to mitigate the effects brought by the Non-IID dataset.**

***Index Terms*—Computer architecture, federated learning, machine learning, edge computing, optimization.**

I. Introduction

COMPUTER architecture has played a crucial role in the development of machine learning (ML) by enabling the efficient processing of large-scale data through innovations in parallel computing, multi-core processors, and specialized hardware such as Graphics Processing Units (GPUs) and Tensor Processing Units (TPUs) [1]–[3]. While traditional computer architecture relies on centralized systems, the rise of distributed systems, the proliferation of Internet of Things (IoT) devices, and increasing concerns over data privacy have driven the shift towards edge computing [4], [5]. Furthermore, within the context of edge computing, Federated Learning (FL) has emerged as a paradigm of distributed machine learning, enabling decentralized clients to collaboratively train a global model using their own private data, thereby addressing the issues of data silos and enhancing user privacy [6]–[9].

In FL, the clients' availability and decentralization bring great communication overhead to the network, and it can be reduced by increasing the local computing [10]. The local computing of FL generally adopts the optimization method of Stochastic Gradient Descent (SGD) [11], which requires the Independent and Identically Distributed (IID) data to ensure the unbiased estimation of the global gradient in the training process [12]–[17]. Since the clients' local data are obtained from the local environment and usage habits, the generated datasets are usually Non-IID due to differences in size and distribution [18]. The Non-IID datasets and multiple local SGD iterations introduce a drift (referred to as 'client-drift') to the global model in each communication round, which significantly reduces the FL performance and stability in the training process, thus requiring more communication rounds to converge or even fail to converge [19]–[21]. Therefore, it becomes a key challenge to accelerate the convergence of FL by mitigating the negative impact of 'client-drift'.

Broaden the perspective beyond FL, here is an illuminating statement: only a significant portion of the training samples play an important role in the generalization performance of the model, and thus dataset pruning can be used to construct the smallest subset from the entire training data as a proxy training

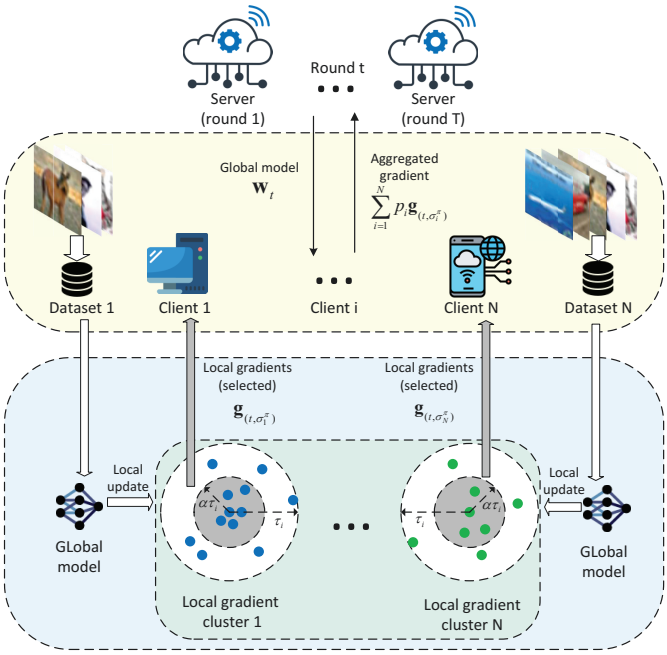


Fig. 1. System architecture for herding local gradients in FL; The yellow segment represents the traditional FL framework, while the green segment illustrates the innovative structure of the BHerd strategy.

Manuscript received. *(Ping Luo and Xiaoge Deng contributed equally to this work. Corresponding author: Tao Sun.)*

Ping Luo, Ziqing Wen, Tao Sun, Dongsheng Li are with the National Key Laboratory of Parallel and Distributed Computing, College of Computer Science and Technology, National University of Defense Technology, Chang-Sha, 410073, China (e-mail: luoping@nudt.edu.cn; zqwen@nudt.edu.cn; suntao.saltfish@outlook.com; dsli@nudt.edu.cn).

Xiaoge Deng is with the Intelligent Game and Decision Lab, Beijing, 100091, China (e-mail: dengxg@ustc.edu).

TABLE I
SUMMARY OF MAIN NOTATIONS

Notation	Interpretation
$F(\boldsymbol{w})$	Global loss value
$F_i(\boldsymbol{w})$	Local loss value on client i
α	The proportion of selected local gradients
E	Local epoch per round for all clients
B	Batch size of local training dataset
N	Number of clients participating in training
η	Learning rate
p_i	Aggregate weight on client i
t	Index of global round
T	Total global round
λ	Index of local iteration
τ_i	Upper bound for the value of λ on client i
$\boldsymbol{w}_{(t,i)}^{\lambda}$	Local model at t-th round, i-th client and λ-th iteration
$\boldsymbol{w}_t$	Global model at t-th round
$\boldsymbol{w}^*$	Optimal model parameters
σ_i	Batch sorting of training samples on the i-th client
σ_i^{π}	Selected batch order of training samples on the i-th client
$\nabla F_i(\boldsymbol{w}_{(t,i)}^{\lambda})$	Gradient of $\boldsymbol{w}_{(t,i)}^{\lambda}$ on training samples
$\boldsymbol{\mu}_{(t,\sigma_i)}$	The average of the gradients from the local update
$\boldsymbol{g}_{(t,\sigma_i^{\pi})}$	The sum of the gradients after Equation (4)
$\nabla F_i(\boldsymbol{w}_t)$	The average of the gradients obtained from the $\boldsymbol{w}_t$
$\nabla F(\boldsymbol{w}_t)$	The global gradient obtained by aggregating $\nabla F_i(\boldsymbol{w}_t)$
L	Constant for Lipschitz smooth in Assumption 1
β	Constant for Differences in Assumption 2
ς	Constant for Bounded gradient error in Assumption 3
γ	Constant for PL condition in Assumption 4

set without significantly sacrificing the model's performance [22], [23]. Translating this concept to FL yields an innovative insight: within a Non-IID dataset, only a subset of samples contribute significantly to the convergence of the global model. By isolating and utilizing the gradients derived from training on these samples, we can prune the gradients that contribute to 'client drift', with the remaining gradients referred to as 'beneficial gradients'. These 'beneficial gradients' are then employed for global updates, thereby enhancing the model's generalization capabilities and accelerating its convergence. Hence, the primary challenge addressed in this paper is the design of an effective methodology for selecting these beneficial gradients, along with a principled analysis of its feasibility and effectiveness in accelerating FL.

In this paper, we first propose a new strategy called BHerd that directly performs dynamic pruning on the local gradients in FL. In each communication round, BHerd orders local gradients using the herding strategy in GraB [24], and then performs the pruning, which means selecting only the top portion (regarded as 'beneficial gradients') in the permutation of the gradients. Fig. 1 illustrates the prototype system architecture, which comprises the following components: 1) Server-side global updates. During the round of global training, the server disseminates the global model to all participating clients, followed by executing a global update upon receipt of the gradients learned by the clients. 2) Client preparation phase. Upon receiving the global model, the client invokes the local dataset and initiates training preparation in accordance with the predefined parameters. 3) Local update to obtain local gradient sets. During this phase, the client undertakes iterative steps of local updates applied to the global gradient to generate distinct local gradients. 4) Selection of beneficial partial local gradients. For enhanced visualization and comprehension, these gradients are represented on a two-dimensional plane within the figure. Points positioned nearer to the circle's center signify a reduced deviation from the mean gradient, thus receiving a higher ranking. Subsequently, the top-ranked local gradients are selected, constituting the final local outcome.

Compared to the GraB method, the BHerd strategy introduces the following innovations. First, we extend the core operation of GraB sorting to local gradients within the FL framework. Unlike GraB, which operates on data samples, the BHerd strategy directly manipulates the gradients. Second, while GraB only reorders the data samples, BHerd not only reorders the gradients but also performs pruning to mitigate the impact of drift gradients. Unlike conventional pruning, BHerd does not discard the samples corresponding to the local gradients. Therefore, each sample has a probability of contributing to the model update throughout the entire training process. Finally, the essence of GraB is to apply the sample sequence from the previous training round to the current round, which introduces delayed information that can hinder model convergence. In contrast, BHerd processes gradients only in the current round, thus avoiding this issue and further accelerating model convergence, as demonstrated in the experimental results in Fig. 3 of Section V-B.

The main contributions of this paper are as follows:

- Conceptually, we relate the local datasets to the local gradient sets generated during the training process in FL. Thus, we extend the 'partial beneficial' concept from the dataset domain, as well as pruning techniques, to the gradient sets.
- Algorithmically, we store the local gradients from the FL training process as permutations, then use the herding method to reorder them and select only the top portion for global aggregation. We name this method BHerd and give the specific algorithm on the FedAvg framework.
- Theoretically, we prove the convergence of BHerd. Our theoretical results show that the FedAvg algorithm is a special case of the BHerd algorithm, and BHerd can control the rate of convergence by controlling the proportion of gradients selected.
- Empirically, We evaluate the general performance of the BHerd algorithm via extensive experiments on general public datasets, which confirms that the BHerd algorithm provides efficient performance in both IID and Non-IID datasets. We also analyze the workings that make the BHerd algorithm effective from the perspective of the gradient distribution, while considering the effects of various hyperparameters on BHerd.

The main notations in this paper are summarized in Table I. This paper is organized as follows. The basic definition of FL and the formalization of the Herding approach are summarized in Section III. The BHerd algorithm and its convergence analysis are presented in Section IV. Experimental results are shown in Section V. The conclusion is presented in Section VI.

II. Related Work

As an integral component of modern computer architecture, edge computing plays a crucial role in accelerating the convergence of Federated Learning (FL). Edge systems are characterized by decentralized data, heterogeneous devices, and limited bandwidth, which pose both unique challenges and opportunities for optimizing FL. The proposed BHerd strategy extends our previous work, AAFL [25], by incorporating additional hardware-aware and architectural adaptations. In particular, AAFL suffers from convergence bottlenecks when faced with Non-IID data distributions in heterogeneous device environments. To achieve faster convergence in FL, it is essential to develop strategies that focus on optimizing both local model training and local data distribution.

A common strategy is adapting the model to local tasks, which creates a better initial model by local fine-tuning. The scheme proposed in [26] makes the local model have a better initial global model by using Model-Agnostic Meta-Learning (MAML) [27]. As an extension, [28] proposes a federated meta-learning formulation using Moreau envelopes. Besides, multi-tasking is proposed to solve statistical challenges in FL environments by finding relationships between clients' data such that similar clients learn similar models [29]–[31]. With the idea of extracting information from large models to small models, knowledge distillation has also been generalized to optimization strategies for FL [32]. For example, a distillation framework for model fusion with robustness is proposed in [33], using unlabeled data output from client models to train a central classifier that flexibly aggregates heterogeneous models of clients. [34] proposed a domain-adaptive federated optimization method that aligns the learned representations of different clients with the data distribution of the target clients and uses feature decomposition to enhance knowledge transfer.

In addition to the optimization of FL strategies by adapting the model, the broader focus aims at improving data distribution from an optimization perspective of the data, thus effectively improving the generalization performance of the global model [35]. Based on this strategy, there is a mean-augmented FL strategy [36], which is inspired by data augmentation methods [37] and allows each client to average the data after exchanging updated model parameters. The averaged data are sent to each client and used to reduce the degree of local data distribution imbalance. Moreover, other related studies treat each client as a domain, and data augmentation is applied to each client's data by selecting transformations related to the overall distribution to generate similar data distributions [38], [39]. Rather than modifying data directly, some schemes indirectly improve data distribution through client-side selection. For example, the FL process can be stabilized and sped up by describing the data distribution on the client through the uploaded model parameters, thus the best subsets of clients are intelligently selected in each communication round [40]. Similarly, there is a scheme to evaluate the class distribution without knowing the original data, which uses a client selection strategy oriented towards minimizing class imbalance [41]. The method proposed in [42] creates a globally shared subset of data to obtain better learning performance in the case of Non-IID data distributions. Besides, there are some schemes that share part of local data with the central server for mitigating the negative impact of Non-IID data [43], [44].

Comparatively, some studies have chosen not to change the data domain, but to tune the local dataset. FedShuffle performs a random reshuffling (RR) operation on the dataset at the beginning of each local epoch and analyzes its convergence theoretically [45] [46]. GraB first formulated an example-ordering framework named herding and answered affirmatively that SGD with herding converges faster than RR [24]. When mapping local data to local gradients via model parameters, it can be found that the local gradients influence the bias of the global aggregation. So some studies have optimized the convergence process of FL by adjusting these local gradients. AAFL allows devices with different performances to have a heterogeneous number of local gradients in each communication round [25]. FedNova rescales the local gradient and the global gradient, which reduces the 'client-drift' caused by Non-IID data to the global update [47]. SCAFFOLD uses control variates (variance reduction) to correct for the 'client-drift' in its local gradients [48]. FedDyn dynamically adjusts the local models' gradients during training by considering the data distribution and model divergence [49].

Through these researches, we identified a pathway to accelerate FL convergence: BHerd that mitigates the influence of Non-IID data through modifications to the local gradient. In contrast to previous studies that focus on model adjustments, BHerd requires no additional modifications to the model architecture. Compared to approaches that manipulate training data, BHerd introduces no extra data processing. Similarly, unlike methods that adaptively tune local training hyperparameters, BHerd does not involve additional parameter adjustments. Therefore, BHerd offers a novel perspective for improving FL. We will introduce the relevant concepts of BHerd in the next section.

III. Definitions of BHerd

In this section, we introduce some related definitions of the BHerd strategy and discuss the effectiveness in the FL framework. For convenience, we assume that all vectors are column vectors in this paper. We use $\boldsymbol{w}$ to denote the model parameters and use $\|.\|$ to denote the L_2 norm. It should be mentioned that we use mini-batch gradient descent with batch size $B \geq 1$, so the differentiable (loss) function $F : \mathbb{R}^d \to \mathbb{R}$ is minimized on a set of data batch $\boldsymbol{x}$. As the first step, we need to get the local gradients of each client on its data batch.

A. Collect Local Gradients

First, we begin by introducing the relevant definitions in the process of FL local model training. Assume that we have N clients with local train datasets $\mathcal{D}_1, \mathcal{D}_2, ..., \mathcal{D}_i, ..., \mathcal{D}_N$, where i denotes the client index and the training sample $\boldsymbol{x}_i \in \mathcal{D}_i$. We then assign weights to individual clients in the traditional approach (FedAvg [10]), $p_i := \frac{|\mathcal{D}_i|}{|\mathcal{D}|}$, where $|.|$ denotes the size of the set and $|\mathcal{D}| = \sum_{j=1}^{N} |\mathcal{D}_j|$ is the size of the total training dataset. In the FL update rules, the training process

has T ($T \geq 1$) communication rounds, and the server has global model parameters $\boldsymbol{w}_t$, where $t = 0, 1, 2, ..., T$ denotes the round index. In each round, each client i trains the local dataset for E epochs, thus has $\tau_i = E\frac{|\mathcal{D}_i|}{B}$ ($\tau_i \geq 1$) local SGD iterations (number of batches) and its local model parameters $\boldsymbol{w}_{(t,i)}^{\lambda}$, where $\lambda = 0, 1, 2, ..., \tau_i$ denotes the local SGD iteration index.

When FL begins ($t = 0$), the server initializes the global model parameter $\boldsymbol{w}_1$ and sends it to all clients. At $\lambda = 1$, the local model parameters for all clients are received from the server that $\boldsymbol{w}_{(t,i)}^{\lambda=1} = \boldsymbol{w}_t$. For $1 \leq \lambda \leq \tau_i$, the gradients $\nabla F_i(\boldsymbol{w}_{(t,i)}^{\lambda}; \boldsymbol{x}_{\sigma_i})$ are computed according to the FedAvg local updates that

$$\boldsymbol{w}_{(t,i)}^{\lambda+1} = \boldsymbol{w}_{(t,i)}^{\lambda} - \eta \nabla F_i(\boldsymbol{w}_{(t,i)}^{\lambda}; \boldsymbol{x}_{\sigma_i}), \tag{1}$$

where η is the learning rate, $F_i(\boldsymbol{w})$ is the local loss function, $\boldsymbol{w}_{(t,i)}^{\lambda}$ are the local model parameters, $\boldsymbol{x}_{\sigma_i}$ is the local data batch and $\sigma_i = \{1, 2, ..., \tau_i\}$ is a permutation (ordering) of data batch $\boldsymbol{x}$.

In contrast to traditional FL and the GraB method, the proposed BHerd strategy does not require the Random Reshuffling (RR) [45] operation on local datasets. Because BHerd can consistently identify data samples (mapped as gradients) that contribute to the convergence of the global model, while the RR operation merely reorders the data samples, having minimal impact on BHerd (as intuitively shown in Fig. 6 and Fig. 7). We will use experimental results in Section V-B5 to show that there is little difference between using and not using the RR protocol in BHerd strategies and that not using the RR protocol makes it more convenient for us to analyze our results.

Next, we will sort these gradients according to the idea of the herding algorithm in GraB, and then select the 'beneficial gradients' based on our BHerd strategy.

B. Herding Local Gradients

Formally, in one epoch at client i, the GraB strategy given τ_i gradients $\{\boldsymbol{z}_\lambda\}_{\lambda=1}^{\tau_i} \in \mathbb{R}^d$, any permutation $\sigma_i^\pi = \{\pi_1, \pi_2, ..., \pi_{\tau_i}\}$ minimizing the term (named average gradient error)

$$\max_{k \in \{1,...,\tau_i\}} \Big\| \sum_{\mu=1}^{k} \big(\boldsymbol{z}_{\pi_\mu} - \frac{1}{\tau_i} \sum_{\lambda=1}^{\tau_i} \boldsymbol{z}_\lambda\big) \Big\|, \tag{2}$$

leads to fast convergence.

In our proposed BHerd strategy, for ease of representation, we define

$$\boldsymbol{\mu}_{(t,\sigma_i)} := \frac{1}{\tau_i} \sum_{\lambda=1}^{\tau_i} \nabla F_i(\boldsymbol{w}_{(t,i)}^{\lambda}; \boldsymbol{x}_{\sigma_i}) \tag{3}$$

to denote the average of the gradients that we obtained in Section III-A.

In GraB, the herding operation is used solely to obtain a new sequence σ_i^π, which is then employed to predict the ordering of the test set $\boldsymbol{x}_{\sigma_i^\pi}$ for the next round of training. In contrast, BHerd does not introduce additional rounds or prediction biases. BHerd sorts the gradients collected to be close to the average gradient $\boldsymbol{\mu}_{(t,\sigma_i)}$ and directly picks only the top $\alpha\tau_i, \alpha \in (0, 1]$ (rounding when not an integer) gradients to form a new permutation. Based on these, we rewrite the Equation (2) as

$$\arg\min_{\sigma_i^\pi} \Big\| \sum_{\lambda=1}^{\alpha\tau_i} \Big(\nabla F_i\big(\boldsymbol{w}_{(t,i)}^{\lambda}; \boldsymbol{x}_{\sigma_i^\pi}\big) - \boldsymbol{\mu}_{(t,\sigma_i)} \Big) \Big\|, \tag{4}$$

where $\sigma_i^\pi = \{\pi_1, \pi_2, ..., \pi_{\alpha\tau_i}\}$. The purpose of Equation (4) is to characterize the entire set of gradients in terms of partial of them, thus mitigating the effects of long-tailed gradients and improving the generalization ability of the model. The specific steps are outlined in Algorithm 2. In Section IV and V, we will theoretically analyze and experimentally validate this strategy.

Algorithm 1 General Framework of FedAvg

Input: Total global round T, local iterations τ_i, initialized weight $\boldsymbol{w}_1$, learning rate η.

```
for t = 1, ..., T do
    for i = 1, ..., N do
        for λ = 1, ..., τ_i do
            Compute the local gradient: ∇F_i(w_{(t,i)}^λ; x_{σ_i}) using Equation (1).
        end for
        Sum all local gradients: Σ_{λ=1}^{τ_i} ∇F_i(w_{(t,i)}^λ; x_{σ_i}).
    end for
    Update the global model w_{t+1} using Equation (7).
end for
return w_{T+1}
```

C. Global Aggregation

After herding and selecting the collected local gradients, BHerd sums the gradients in the permutation σ_i^π to get

$$\boldsymbol{g}_{(t,\sigma_i^\pi)} = \sum_{\lambda=1}^{\alpha\tau_i} \nabla F_i\big(\boldsymbol{w}_{(t,i)}^{\lambda}; \boldsymbol{x}_{\sigma_i^\pi}\big). \tag{5}$$

At the end of t-th global round, when all clients have completed this step, $\boldsymbol{g}_{(t,\sigma_i^\pi)}$ is sent to the parameter server-side to complete the aggregation update step as

$$\begin{aligned} \boldsymbol{w}_{t+1} &= \boldsymbol{w}_t - \eta \frac{E}{\alpha} \sum_{i=1}^{N} p_i \boldsymbol{g}_{(t,\sigma_i^\pi)} \\ &= \boldsymbol{w}_t - \frac{\eta}{\alpha} \sum_{i=1}^{N} p_i \boldsymbol{g}_{(t,\sigma_i^\pi)}. \end{aligned} \tag{6}$$

The last item is due to the fact that we set $E = 1$, which is reasonable, i.e., we only consider the case where each client is trained for only 1 epoch since too large or too small an epoch can lead to a bias in the global model and thus affect convergence.

D. BHerd is an extension of FedAvg and effective

For the purpose of comparison with our proposed BHerd strategy, we will introduce here the most primitive FL algorithm: the FedAvg. The logic of FedAvg with distributed SGD is presented in Algorithm 1, which ignores aspects related to the communication between the server and clients [10]. After the previous description of the training process of FL, the FedAvg algorithm becomes clear, and can be formally described as

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta \sum_{i=1}^{N} p_i \sum_{\lambda=1}^{\tau_i} \nabla F_i(\boldsymbol{w}_{(t,i)}^{\lambda}; \boldsymbol{x}_{\sigma_i}), \tag{7}$$

where it can be seen as a special case of our BHerd strategy in the case of $E = 1$ and $\alpha = 1$. According to the definition of Equation (3), the Equation (7) can also be represented as $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta \sum_{i=1}^{N} p_i \tau_i \boldsymbol{\mu}_{(t,\sigma_i)}$. Compared to FedAvg, a natural research question is

How does our BHerd strategy work and is it more effective than FedAvg?

Proposition 1. *According to our definition of global aggregation for the BHerd strategy, its gradient aggregation is equivalent to model parameter aggregation, that is*

$$\boldsymbol{w}_{t+1} = \sum_{i=1}^{N} p_i \boldsymbol{w}_{(t,i)}^{\tau_i+1}. \tag{8}$$

Proof. For details, see Appendix A. □

Equation (8) is another representation of FedAvg's global aggregation, which performs global model parameter updating by aggregating the local model parameters obtained after local updating. Thus we can answer the first half of the question posed: the BHerd strategy works in line with FedAvg and is an extension of it.

For the second half of the problem, we will give the convergence analysis in Section 4 and the experimental validity analysis in Section 5. In this section of the paper, we first give the abstract analysis in a rough way. We first simplify all the operations of the BHerd strategy to the two-dimensional plane, i.e., model parameters $\boldsymbol{w} \in \mathbb{R}^2$, the local gradient $\nabla F_i(\boldsymbol{w}) \in \mathbb{R}^2$ and the global gradient $\nabla F(\boldsymbol{w}) \in \mathbb{R}^2$. Then, we will make an abstraction of their variations according to the training process we defined above. As shown in Fig. 2, we assume that there are two clients, one with 6 local update iterations ($\tau_i = 6$) and the other with 4 local update iterations ($\tau_i = 4$). A permutation of the local gradient $\nabla F_i(\boldsymbol{w}_{(t,i)}^{\lambda}; \boldsymbol{x}_{\sigma_i})$ is obtained on each client i in the order of σ_i. It is then reordered using the herding policy, and the top $\alpha\tau_i$ elements are selected ($\alpha = 0.5$ in Fig 2). Up to this point, we can intuitively see that our proposed herding strategy is able to characterize all the gradient information with a portion of the local gradient. This has the advantage of reducing the impact of the biased gradient $\nabla F_i(\boldsymbol{w}_{(t,i)}^{\lambda}; \boldsymbol{x}_{\sigma_i - \sigma_i^{\pi}})$, which allows us to increase the step size ($\frac{1}{\alpha}$) global gradient to accelerate the convergence. This conclusion can also be easily generalized to $\mathbb{R}^d$-space.

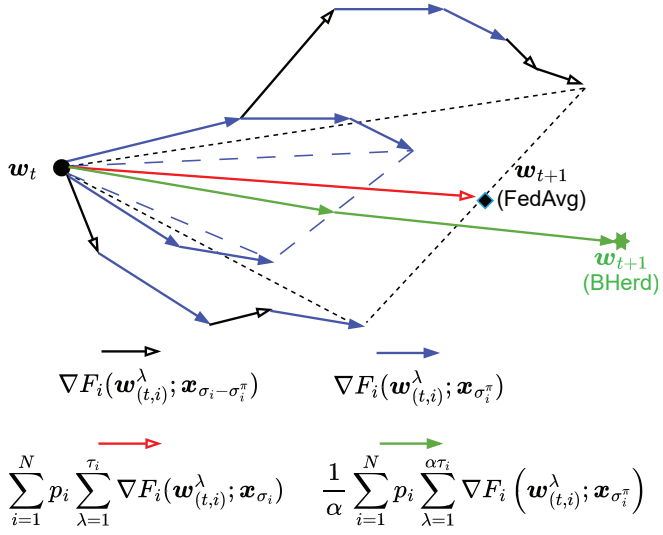


Fig. 2. Analyze the effectiveness of our proposed method in the abstract on a $\mathbb{R}^2$ plane, we set $\alpha = 0.5$.

In the next section, we will apply the proposed herding strategy and the new FL training procedure to design the algorithm and formally analyze its convergence.

IV. The BHerd Algorithm and Convergence Analysis

In this section, based on the definitions of the BHerd strategy obtained in Section III, we first design the BHerd algorithm under the FL framework. Next, we analyze the convergence of the proposed BHerd strategy and get a convergence upper bound in the non-convex hypothesis.

A. BHerd Algorithm

The BHerd's herding steps can be used to minimize Equation (4) that finds a better data permutation σ_i^{π}, and is shown in Algorithm 2 which starts with a group of gradients $\{\boldsymbol{z}_\lambda\}_{\lambda=1}^{\tau}$. After finding the average gradient $\frac{1}{\tau}\sum_{\mu=1}^{\tau} \boldsymbol{z}_\mu$, all the gradients subtract this average gradient to get the deviations, which are assigned as the new gradients to $\{\boldsymbol{z}_\lambda\}_{\lambda=1}^{\tau}$ (Line 1 of Algorithm 2).

To obtain the new order σ_i^{π}:$\{\pi_1, \pi_2, \ldots, \pi_{\alpha\tau_i}\}$, Algorithm 2 needs to select $\alpha\tau_i$ gradients from the set $\{\boldsymbol{z}_\lambda\}_{\lambda=1}^{\tau}$ without repetition, aiming to minimize the value of $\left\|\boldsymbol{z}_{\pi_1} + \cdots + \boldsymbol{z}_{\pi_{\alpha\tau_i}}\right\|$. The specific approach is as follows: at each iteration, a single gradient $\boldsymbol{z}_\mu$ is chosen to minimize the $\|\boldsymbol{s} + \boldsymbol{z}_\mu\|$ value, where $\boldsymbol{s}$ represents the cumulative of the selected gradients (Line 4 and 5 of Algorithm 2). After selecting a gradient $\boldsymbol{z}_\mu$, it is removed from the gradient set $\{\boldsymbol{z}_\lambda\}_{\lambda=1}^{\tau}$, and the order π is reordered according to the selection index μ Line 5 of Algorithm 2.

Based on the herding steps in Algorithm 2, we can clearly get the BHerd algorithm under the FL framework: when training the local model for an epoch, we herd all the local gradients in the current global round and take the first a items of them to form a new permutation, and we send all the local gradients in this new permutation to the parameter server

Algorithm 2 The herding steps in BHerd

Input: A group of gradients $\{\boldsymbol{z}_\lambda\}_{\lambda=1}^{\tau}$, the proportion α.

1: Center all the vectors: $\boldsymbol{z}_\lambda \leftarrow \boldsymbol{z}_\lambda - \frac{1}{\tau}\sum_{\mu=1}^{\tau}\boldsymbol{z}_\mu, \forall\lambda \in [\tau]$.
2: Initialize an arbitrary σ_i^π, running partial sum: $\boldsymbol{s} \leftarrow \boldsymbol{0}$, candidate set $\sigma_i \leftarrow \{1, ..., \tau\}$.
3: **for** $\lambda = 1, ..., \alpha\tau$ **do**
4: Iterate through σ_i and select $\boldsymbol{z}_\mu$ from σ_i that minimizes $\|\boldsymbol{s} + \boldsymbol{z}_\mu\|$.
5: Remove μ from σ_i, update partial sum and order: $\boldsymbol{s} \leftarrow \boldsymbol{s} + \boldsymbol{z}_\mu$; $\sigma_i^\pi(\lambda) \leftarrow \mu$.
6: **end for**
7: **return** σ_i^π

Algorithm 3 BHerd algorithm under FL framework

Input: Total global round T, local iterations τ_i, initialized order σ_i^π at client i, initialized weight $\boldsymbol{w}_1$, learning rate η.

1: **for** $t = 1, ..., T$ **do**
2: **for** $i = 1, ..., N$ **do**
3: **for** $\lambda = 1, ..., \tau_i$ **do**
4: Compute the local gradient: $\nabla F_i(\boldsymbol{w}_{(t,i)}^\lambda; \boldsymbol{x}_{\sigma_i})$ using Equation (1).
5: Store the gradient: $\boldsymbol{z}_\lambda \leftarrow \nabla F_i(\boldsymbol{w}_{(t,i)}^\lambda; \boldsymbol{x}_{\sigma_i})$.
6: **end for**
7: Generate new order: $\sigma_i^\pi \leftarrow Herding(\{\boldsymbol{z}_\lambda\}_{\lambda=1}^{\tau})$.
8: Sum gradients: $\boldsymbol{g}_{(t,\sigma_i^\pi)}$ using Equation (5).
9: **end for**
10: Update the global model $\boldsymbol{w}_{t+1}$ using Equation (6).
11: **end for**
12: **return** $\boldsymbol{w}_{T+1}$

for global update. This herding-gradient approach is formally described in Algorithm 3. Algorithm 3 can be considered as adding the herding operation (Algorithm 2) of the local gradients to the FL framework (Algorithm 1), where the local gradient $\nabla F_i(\boldsymbol{w}_{(t,i)}^\lambda; \boldsymbol{x}_{\sigma_i})$ corresponds to the vector $\{\boldsymbol{z}_\lambda\}_{\lambda=1}^{\tau}$ in Algorithm 2. After obtaining a new permutation σ_i^π, we can compute the corresponding $\boldsymbol{g}_{(t,\sigma_i^\pi)}$ by using Equation (5), and finally use the new aggregation rule Equation (6) to update the global model (Line 7, 8 and 10 in Algorithm 3).

B. Convergence Analysis

In BHerd, for each global model $\boldsymbol{w}$, there is its corresponding loss function $F(\boldsymbol{w})$. The objective of FL is to find the optimal global model $\boldsymbol{w}^*$ to minimize $F(\boldsymbol{w})$, which is formally described as

$$\boldsymbol{w^*} = \arg\min F(\boldsymbol{w}). \tag{9}$$

For the global loss function $\boldsymbol{w}$, it is also necessary to define its global gradient. The following will describe how we obtain this global gradient in BHerd.

Definition 1. *At each client i in round t, we train the global model parameter $\boldsymbol{w}_t$ with the local datasets, with the difference that this model parameter will not be involved in the local update of Equation* (1)*, but will remain fixed at round t and across all clients.*

$$\nabla F_i(\boldsymbol{w}_t) := \frac{1}{\tau_i}\sum_{\lambda=1}^{\tau_i}\nabla F_i(\boldsymbol{w}_t; \boldsymbol{x}_{\sigma_i}).$$

Then weighted sum $\nabla F_i(\boldsymbol{w}_t)$ at each client to obtain the global gradient $\nabla F(\boldsymbol{w}_t)$.

$$\nabla F(\boldsymbol{w}_t) := \sum_{i=1}^{N} p_i \nabla F_i(\boldsymbol{w}_t).$$

Definition 1 is commonly used definition in the study of FL [10]. For further analysis, we use some assumptions for non-convex optimization.

Assumption 1 (Lipschitz smooth)**.** *For any $w, v \in \mathbb{R}^d$, it holds that*

$$\|\nabla F(\boldsymbol{w}) - \nabla F(\boldsymbol{v})\| \le L\|\boldsymbol{w} - \boldsymbol{v}\|.$$

Assumption 2 (Differences in model parameters)**.** *In ground $t \in [1, T]$, if we define $\Delta_t := \max_{i,\lambda}\|\boldsymbol{w}_t - \boldsymbol{w}_{(t,i)}^\lambda\|$ for all clients $i \in [1, N]$ and local iteration $\lambda \in [2, \tau_i + 1]$, it exists $\beta \in (0, 1]$ that*

$$\beta\Delta_t \le \|\boldsymbol{w}_t - \boldsymbol{w}_{t+1}\|.$$

Assumption 3 (Bounded gradient error)**.** *For any $t \in [1, T]$ and $i \in [1, N]$, it exists $\varsigma > 0$ and $\xi > 0$ that*

$$\frac{1}{\tau_i}\sum_{\lambda=1}^{\tau_i}\|(\nabla F_i(\boldsymbol{w}) - \nabla F_i(\boldsymbol{v}))\|^2 \le \varsigma^2,$$

$$\sum_{i=1}^{N} p_i\|\nabla F_i(\boldsymbol{w}_t) - \nabla F(\boldsymbol{w}_t)\|^2 \le \xi^2.$$

Assumption 4 (PL condition)**.** *We say the loss function F fulfills the Polyak-Lojasiewicz (PL) condition if there exists $\gamma > 0$ such that for any $\boldsymbol{w}_t \in \mathbb{R}^d$,*

$$\frac{1}{2}\|\nabla F(\boldsymbol{w})\|^2 \ge \gamma\big(F(\boldsymbol{w}) - F(\boldsymbol{w}^*)\big),$$

where $F(\boldsymbol{w}^*) = \inf_{\boldsymbol{v}\in\mathbb{R}^d} F(\boldsymbol{v})$.

The symbol of $\|.\|$ denotes the L norm of vector, and we use the default L_2 norm in this paper. Assumption 1, 3 and 4 are commonly used assumptions in the study of SGD and convex optimization [24], while Assumption 2 is specifically required for our BHerd strategy. Assumption 1 ensures that the function's variation is smooth, which facilitates the analysis of convergence speed. Assumption 2 reflects the phenomenon of 'client drift' in FL and helps analyze convergence by restricting the variation of model parameters through β. Assumption 3 allows for a certain level of gradient deviation in each client's computation, but this deviation does not grow unbounded. Assumption 4 guarantees that even if the objective function is not strictly convex, it still ensures the existence of a global optimal solution and the convergence of the gradient descent method.

Before we analyze the upper bound on the convergence of the BHerd algorithm, we introduce some of the following Lemmas. These lemmas are derived under the steps of the BHerd algorithm, based on the aforementioned assumptions.

Lemma 1. *We denote the maximal number of local iterations for all clients by $\tau_{max} := \max_{i\in[1,N]} \tau_i$. If $\alpha\eta L\tau_{max} \leq 1$ with $\alpha \in (0, 1]$ holds and Assumption 1 and 2 hold, then*

$$\frac{1}{T}\sum_{t=1}^{T} \|\nabla F(\boldsymbol{w}_t)\|^2 \leq \frac{2\left(F(\boldsymbol{w}_1) - F(\boldsymbol{w}^*)\right)}{\alpha\eta\tau_{max}T} + \frac{L^2}{T}\sum_{t=1}^{T}\Delta_t^2.$$

Proof. For details, see Appendix B. □

In Lemma 1, with both the dataset and the model structure already determined, the loss value of its optimal model parameter $\boldsymbol{w}^*$ is uniquely determined. This means that $F(\boldsymbol{w}_1) - F(\boldsymbol{w}^*)$ is fixed, and the magnitude of the value is determined by the initialized model parameter $\boldsymbol{w}_1$. Parameters α, η, and T are manually set fixed values. Parameter τ_{max} is a fixed value only with respect to the total dataset size $|\mathcal{D}|$ and the number of clients N. Parameter L is the smooth upper bound that we defined in Assumption 1, and is also fixed in the actual FL training.

Next, Lemma 2 will prove the relationship between the upper bound of $\sum_{t=1}^{T}\Delta_t^2$ and the global gradient $\nabla F(\boldsymbol{w}_t)$.

Lemma 2. *Under Assumptions 1, 2 and 3, the following inequalities hold:*

$$\Delta_{t+1}^2 \leq \frac{4\eta^2\tau_{max}^2}{\beta^2}\left(2\varsigma^2 + \xi^2 + \|\nabla F(\boldsymbol{w}_t)\|^2\right),$$

and

$$\frac{4\eta^2\tau_{max}^2}{\beta^2}\left(2\varsigma^2 + \xi^2\right),$$

and finally

$$\sum_{t=1}^{T}\Delta_t^2 \leq \frac{4\eta^2\tau_{max}^2 T}{\beta^2}\left(2\varsigma^2 + \xi^2\right) + \frac{4\eta^2\tau_{max}^2}{\beta^2}\sum_{t=1}^{T-1}\|\nabla F(\boldsymbol{w}_t)\|^2.$$

Proof. For details, see Appendix B. □

According to the definition of ς in Assumption 3, we can always find a suitable fixed value of ς in FL training. Then, it is only sufficient to carry over the result in Lemma 2 to Lemma 1, which leads to Theorem 1.

Theorem 1. *In the BHerd strategy, under Assumptions 1, 2 and 3, if the learning rate η and the total global round T fulfill $\eta \leq \frac{\sqrt{2}\beta}{4L\tau_{max}}$ and $T \to \infty$, then it converges at the rate*

$$\frac{1}{T}\sum_{t=1}^{T} \| \nabla F(\boldsymbol{w}_t) \|^2 \leq \frac{4\big(F(\boldsymbol{w}_1) - F(\boldsymbol{w}^*)\big)}{\alpha\eta\tau_{max}T} + 2\varsigma^2 + \xi^2.$$

Proof. For details, see Appendix B. □

From Theorem 1, the right-hand side of the inequality is a fixed value when the appropriate values of T and η are selected, proving that our algorithm converges below a definite value.

Theorem 2. *Under Assumption 1, 2, 3 and 4 (PL condition), if $\eta \leq \frac{\sqrt{2}\beta}{4L\tau_{max}}$ and $T \to \infty$ hold, then the BHerd strategy converges at the upper bound*

$$F(\boldsymbol{w}_T) - F(\boldsymbol{w}^*) = \mathcal{O}\Big(\frac{F}{T}\Big),$$

where $\mathcal{O}$ swallows all other constants and $F := F(\boldsymbol{w}_1) - F(\boldsymbol{w}^*)$.

Proof. For details, see Appendix B. □

In the next section, we design experiments on the general dataset to verify the general effectiveness of the proposed algorithm.

V. Performance Analysis

To evaluate the proposed algorithm, the experiments focused on the qualitative and quantitative analysis of general properties under the setup of our prototype system and the simulation of IID and Non-IID datasets.

A. Setup

We first evaluated the general performance of the proposed algorithm, thus we conducted experiments on a networked prototype system with five clients. In AAFL [25], the prototype system was architected as a parallel distributed framework composed of five Raspberry Pi 4B nodes and a central host interconnected within a local area network. While this hardware configuration closely emulates practical edge computing environments, the implementation of BHerd employs simulated clients on a high-performance host machine. This adjustment is made to ensure the stable execution of the core sorting threads, which were unable to run reliably on Raspberry Pi 4B devices due to their constrained computational capabilities. The prototype system consists of five parallel distributed processes (clients) and one process for parameter initialization, distribution, and handling (server), which are implemented on an instance configured with a 24-core Intel(R) i9-13900K 5.8GHz CPU, 32GB memory and an NVIDIA GeForce RTX 4070 Ti GPU. The server has an aggregator and implements global update and offline herding steps of FL, and the five clients have model training with local datasets and different simulated dataset distributions of IID and Non-IID. We set up that in the BHerd algorithm, each client trains the local dataset for 1 epoch ($E = 1$) in a global update round, with a total of 500 global rounds ($T = 100$).

1) Baselines: We compare with the following baseline approaches:

- **Centralized SGD.** The entire training dataset is stored on a single client, and the model is trained directly on that client using a standard SGD and random reshuffling protocol [45] with $E = 100$.
- **FedAvg.** The standard FL approach [10] that all clients use SGD and random reshuffling protocol with local epoch $E = 1$ and global round $T = 100$.
- **GraB-FedAvg.** The online Gradient Balancing (GraB) [24] operation replaces the herding operation in the BHerd algorithm with local epoch $E = 1$ and global

round $T = 100$. In the GraB operation, when the current local gradient is received, it is roughly ordered based on the average of all the previous gradients to reduce the storage overhead and computational complexity.

- **FedNova.** A novel FL approach [47] that all clients use SGD and random reshuffling protocol with local epoch $E = 1$ and global round $T = 100$. Instead of ordering and selecting the local gradients, the FedNova algorithm simply performs an averaging operation and then scales the global gradients.
- **SCAFFOLD.** A novel FL approach [48] that all clients use SGD and random reshuffling protocol with local epoch $E = 1$ and global round $T = 100$. SCAFFOLD uses control variates (variance reduction) to correct for the 'client-drift' in its local updates.
- **FedDyn.** A novel FL approach [49] that all clients use SGD and random reshuffling protocol with local epoch $E = 1$ and global round $T = 100$. FedDyn proposes a dynamic regularizer for each device at each round so that the global and device solutions are aligned at the limit.

A detailed implementation of the GraB-FedAvg, FedNova, and SCAFFOLD algorithms can be found in Appendix C-A. For a fair comparison, we set the result of the first baseline as the standard model under its epoch budget and use this model for evaluating the convergence of other trained models.

2) Model and Datasets: We evaluate the training of three different models on three different datasets, which represent both small and large models and datasets. The models include squared-SVM[1] (we refer to as SVM in short in the following), deep convolutional neural network (CNN)[2] and recurrent neural network (RNN) [3].

SVM is trained on the original MNIST (which we refer to as MNIST in short in the following) dataset [50], which contains the gray-scale images of 7×10^4 handwritten digits (6×10^4 for training and 10^4 for testing).

CNN is trained using SGD on the CIFAR-10 dataset [51], which includes 6×10^4 color images (5×10^4 for training and 10^4 for testing) associated with a label from 10 classes. A separate CNN model is trained on each dataset, to perform multi-class classification among the 10 different labels in the dataset.

The RNN model is trained using SGD on the IMDB dataset [52], which contains 2.5×10^4 movie reviews for training and 2.5×10^4 for testing. Each review is associated with a binary sentiment label, either positive or negative. The model is trained to perform binary classification, distinguishing between positive and negative sentiment in the reviews.

3) Simulation of Dataset Distribution: To simulate the dataset distribution, we set up two different Non-IID cases and a standard IID case.

- **Case 1** (IID): Each data sample is randomly assigned to a client, thus each client has a uniform (but not full) information.
- **Case 2** (Non-IID): The first half of data samples are distributed to the first half of the clients as in **Case 1**. The remaining samples are assigned to the second half of the clients, ensuring that each client in this group has a unique label.

4) Training and Control Parameters: At the beginning of our experiments, we set the size of the total training dataset $|\mathcal{D}| = 6\times10^4$ for MNIST, $|\mathcal{D}| = 5\times10^4$ for CIFAR and $|\mathcal{D}| = 2.5 \times 10^4$ for IMDB. The batch size and training epoch are constant with the value of $B = 100$ and $E = 1$, respectively. It is important to mention that we assume in this paper that all clients participate in training, and this assumption can easily be generalized to pick a different fraction of clients to participate in each round of training.

When fastening the number of clients N, the local data size $|\mathcal{D}_i|$ will have different fixed values in the different cases, and then we can get a fixed value for $\tau_{(t,i)} = \lfloor E\frac{|\mathcal{D}_i|}{B} \rfloor$ to the clients for each round (except for the first baseline with $\tau = \lfloor E\frac{|\mathcal{D}|}{B} \rfloor$ in one client). Unless otherwise specified, we set the control parameter α to a fixed value $\alpha = 0.5$ for all rounds. We manually select the total round $T = 100$. As for the learning rate fixed, we set $\eta = 2 \times 10^{-4}$ for SVM and CNN, and $\eta = 2 \times 10^{-3}$ for RNN, which is acceptable for our learning process. It is worth noting that, for the purpose of facilitating a straightforward comparison of the experimental results, we used a fixed learning rate instead of a decaying learning rate. This choice leads to oscillations in the prediction accuracy as the model approaches convergence when using the SGD optimizer. Except for the instantaneous results, the others are the average results of 10 independent experiment runs.

B. Results

1) Performance comparison between BHerd and baseline approaches: In our first set of experiments, the SVM, RNN, and CNN models were trained on the prototype system with **Case 1-2** and five clients ($N = 5$). In the **Case 1-2**, the results $\boldsymbol{w}_k$ of each round will be calculated on the test dataset with the prediction accuracy values which we refer to as 'accuracy' in the following.

We record the accuracy of the SVM, RNN and CNN classifiers with BHerd algorithm (with $\alpha = 0.5$), and compare them to baseline approaches, where the centralized case only has one optimal value as the training result. The curves of accuracy values as a function of round T are shown in Fig. 3a, and we show a flat line across different rounds for ease of determining whether the other methods converge to optimum or not. After 100 rounds ($T = 100$) of training, the model accuracy (abbreviated as Acc) is recorded in Table II.

The initial objective of this study is to ascertain the differential efficacy in the convergence rates of FL models facilitated by the GraB sorting algorithm delineated in [24] as compared to the BHerd strategy. As shown in Fig. 3, the BHerd algorithm markedly enhances the speed at which convergence is attained, as well as the ultimate predictive precision of the model, sur-

[1]The squared-SVM has a fully connected neural network, and outputs a binary label that corresponds to whether the digit is even or odd.

[2]The CNN has two $5 \times 5 \times 32$ convolution layers, two 2×2 MaxPoll layers, a 1568×256 fully connected layer, a 256×10 fully connected layer, and a softmax output layer with 10 units.

[3]The RNN model has an embedding layer with a vocabulary size of 1000, an embedding dimension of 100, two GRU layers with 100 hidden units, two fully connected layers: a 200×50 layer with ReLU activation and a 50×2 layer with a softmax output for binary classification.

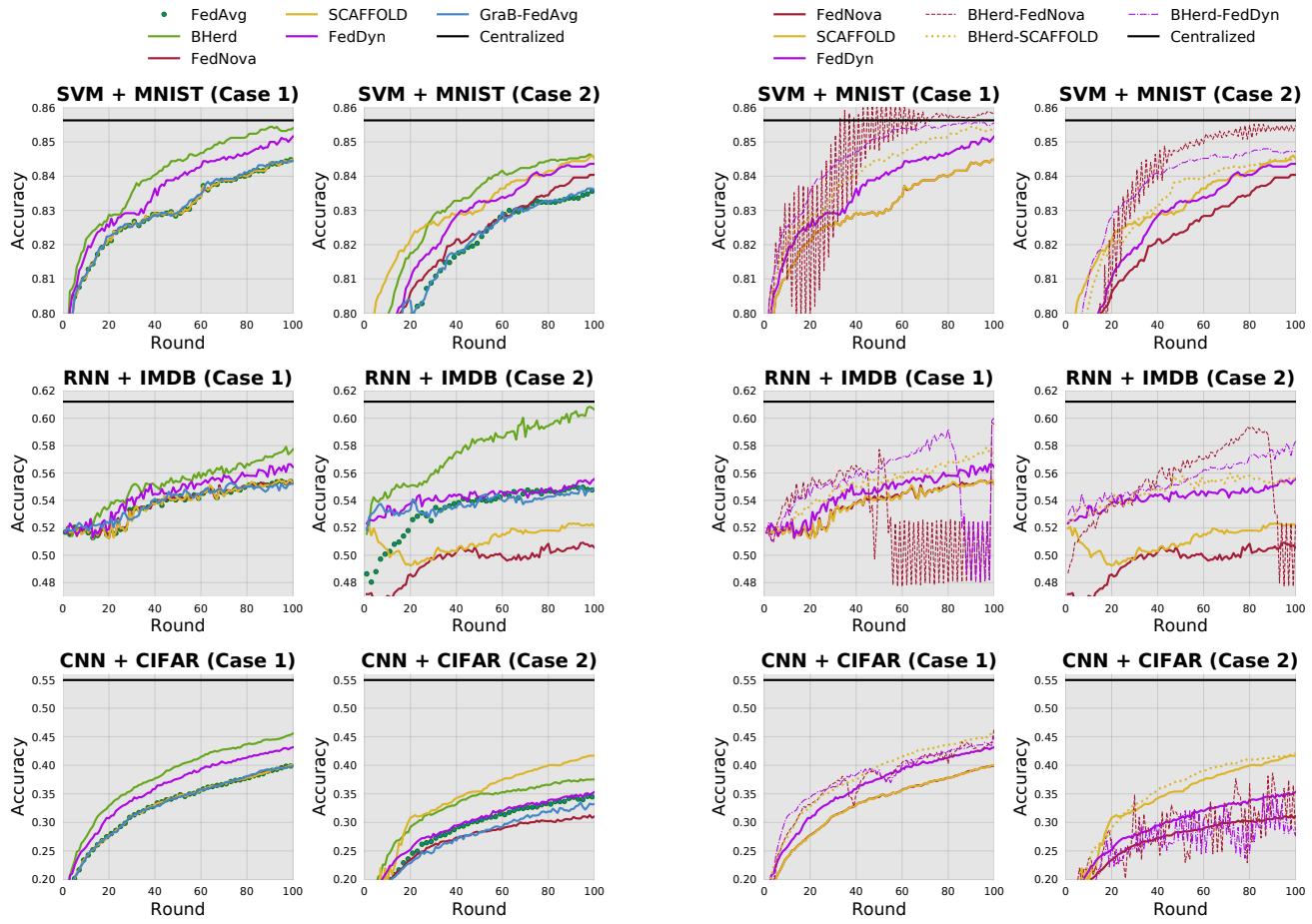


Fig. 3. Classification accuracy values in **Case 1-2**; The black line delineates the outcome of the centralized gradient descent previously discussed, signifying the model's convergence optimum; The green circles signify the conventional FedAvg algorithm, serving as the optimization benchmark; The blue dashed line represents the GraB-FedAvg algorithm, while the other curves correspond to the basic algorithm and its extension utilizing the BHerd strategy, respectively.

passing the performance of the traditional FedAvg algorithm. Conversely, despite the GraB algorithm's superior performance in computational complexity and storage requirements, it falls short in enhancing the convergence rate of FedAvg. This phenomenon arises from the coarse and somewhat random nature of the ranking in the GraB algorithm, in contrast to the precise and systematic ranking in the BHerd algorithm (see Fig. 6). Additionally, a comparison of the GraB and FedAvg curves suggests that as the selected 'beneficial gradients' in the BHerd algorithm become more random, its performance converges toward that of FedAvg.

We observed oscillations in the FedAvg accuracy curve in both the SVM+MNIST and RNN+IMDB scenarios. This is because we used a fixed learning rate η and the SGD optimizer, which is typical for loss functions with multiple saddle points [53]. Consequently, the accuracy curves of the BHerd algorithm designed on top of FedAvg, as well as other baseline approaches, also exhibit oscillations. However, even under unstable convergence conditions, our BHerd algorithm achieves the fastest model convergence across all models and datasets in **Case 1-2**, except for the CNN+CIFAR scenario in **Case 2**, where the SCAFFOLD algorithm performs better.

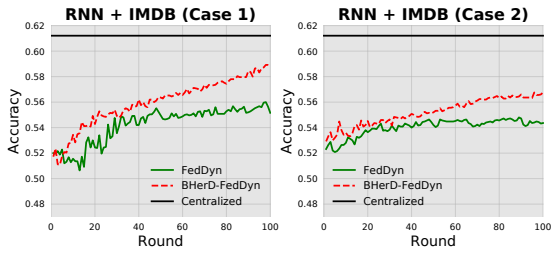


Fig. 4. Classification accuracy values in **Case 1-2** with decayed learning rate η

To optimize the SCAFFOLD algorithm using the BHerd concept and further investigate the generalizability of the BHerd algorithm, we will extend its application framework in the following sections.

2) BHerd's extension to the popular algorithms: Despite the fact that our theoretical validation of the BHerd strategy was confined within the bounds of the FedAvg framework, the concept of BHerd can be extended to other popular algorithms, such as FedNova, FedDyn and SCAFFOLD algorithms, which have gained prominence in recent years.

TABLE II
ACCURACIES VALUES FOR EACH BASELINE ALGORITHM OBTAINED IN **CASE 1-2** (ACC-C1 AND ACC-C2)

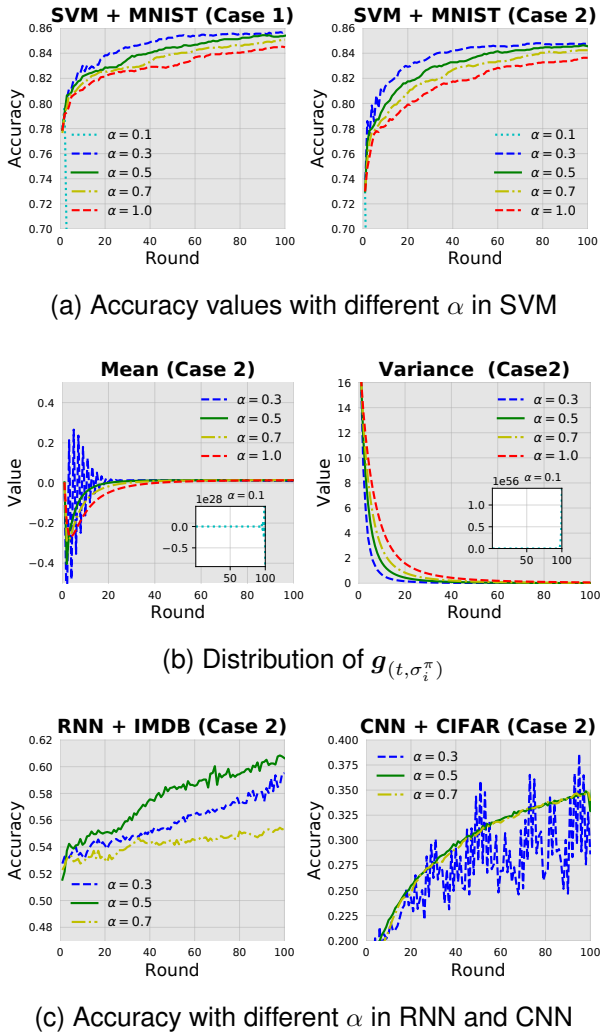
at the end of the set training rounds ($T = 100$).

Model	Algorithm	Acc-C1 (%)	Acc-C2 (%)
SVM MNIST	FedAvg	84.48	83.60
	BHerd	85.40	84.56
	GraB-FedAvg	84.44	83.64
	FedNova	84.48	84.04
	BHerd-FedNova	**85.84**	**85.48**
	SCAFFOLD	84.48	84.52
	BHerd-SCAFFOLD	85.40	84.44
	FedDyn	85.16	84.36
	BHerd-FedDyn	85.52	84.72
RNN IMDB	FedAvg	55.20	54.64
	BHerd	57.72	**60.64**
	GraB-FedAvg	55.24	54.88
	FedNova	55.20	50.56
	BHerd-FedNova	59.56	52.32
	SCAFFOLD	55.16	52.12
	BHerd-SCAFFOLD	57.68	55.64
	FedDyn	56.44	55.56
	BHerd-FedDyn	**60.00**	58.32
CNN CIFAR	FedAvg	40.00	34.44
	BHerd	45.56	37.52
	GraB-FedAvg	39.88	33.20
	FedNova	39.92	31.04
	BHerd-FedNova	**46.48**	27.60
	SCAFFOLD	39.96	41.68
	BHerd-SCAFFOLD	45.56	**42.16**
	FedDyn	43.16	35.32
	BHerd-FedDyn	45.20	27.64

As depicted in Fig. 3b and Table II, the integration of our BHerd strategy into the frameworks of FedNova, FedDyn and SCAFFOLD results in varying degrees of improvement in convergence speeds and final classification accuracy for these algorithms. The FedNova algorithm employs gradient scaling to adjust the step size of global model updates, and our BHerd algorithm further leverages gradient scaling. However, when the scaling factor is excessively large, it amplifies gradient fluctuations, intensifying the optimization jumps and resulting in oscillations in the convergence curve [54]. Therefore, the applicability of BHerd to FedNova is limited and will not be further discussed. In contrast, SCAFFOLD and FedDyn, which focus on correcting both the local and global gradients without significantly altering their respective magnitudes, introduce additional gradient information that acts to dampen the oscillation effect. In the RNN+IMDB scenario, BHerd causes the FedDyn curve to deviate from convergence. This is because the gradient magnitude is typically small near the optimal point, but the fixed learning rate does not dynamically adjust to the gradient magnitude, resulting in model parameter updates overshooting the optimal point. This leads to oscillations and failure to achieve complete convergence.

We further apply a decayed learning rate, reducing η by 50% at the 50th round for both FedDyn and BHerd-FedDyn. As shown in Fig. 5, the decayed learning rate significantly alleviates the convergence deviation and reduces oscillations in the latter part of the curve, which is consistent with our previous conclusion.

Collectively, the BHerd strategy substantiates its efficacy in relation to conventional methodologies, evidenced by its convergence towards enhanced accuracy across standard models, datasets, and data distribution frameworks. Owing to the intricate challenges associated with evaluating CNN and RNN models, this study henceforth narrows its focus to the examination of the SVM model utilizing the MNIST dataset and BHerd-FedAvg framework, aiming to elucidate further on the prototype system's performance. Subsequently, an analysis delineating the correlation between the algorithm's efficiency and the configuration of pertinent hyperparameters will be conducted.



(a) Accuracy values with different α in SVM

(b) Distribution of $\boldsymbol{g}_{(t,\sigma_i^\pi)}$

(c) Accuracy with different α in RNN and CNN

Fig. 5. A detailed analysis of the key parameter α in BHerd.

3) Sensitivity to α and the acceleration mechanism of BHerd: For the same justification as in Section V-B1, we first show the results of running 'SVM+MNIST' and using different values of α (0.1, 0.3, 0.5, 0.7 and 1.0) for the five clients in **Case 1-2**. The quantitative outcomes pertaining to the accuracy are depicted in Fig. 5a. Upon comparative analysis with the 'SVM+MNIST' subplots illustrated in Fig. 3, it is discernible that the convergence trajectory of the algorithm under scrutiny bears resemblance to that of the FedAvg algorithm at $\alpha = 1.0$. This observation suggests that the FedAvg algorithm can be regarded as a particular instantiation of the algorithm proposed herein. Notably, a reduction in the value of α from 1.0 to 0.3 engenders an acceleration in the convergence rate of the BHerd

strategy. Contrariwise, diminishing α further to 0.1 results in the algorithm's failure to converge.

To further investigate the underlying reasons, we analyze the global gradient $\boldsymbol{g}_{(t,\sigma_i^\pi)}$ generated in each training round (determined by Eq. 5). Specifically, we compute the Gaussian distribution of the weight values in the gradient and record the changes in mean and variance over the training rounds. As shown in Fig. 5b, when $\alpha = 1$, the global gradient $\boldsymbol{g}_{(t,\sigma_i^\pi)} = \boldsymbol{\mu}_{(t,\sigma_i)}$. Compared to the curve corresponding to $\boldsymbol{\mu}_{(t,\sigma_i)}$, the accelerated convergence phenomenon in BHerd with $\alpha = 1$ to 0.5 is related to the variance of the global gradient, while the non-convergence and oscillation phenomenon with $\alpha = 0.1$ to 0.3 is associated with the mean. Therefore, we identify the impact of the α parameter on the BHerd algorithm: *to minimize the variance of the global gradient while ensuring that its mean does not deviate from $\boldsymbol{\mu}_{(t,\sigma_i)}$, thereby reducing the negative impact of 'non-beneficial gradients' and accelerating model convergence.*

We also validate the above conclusion in CNN and RNN models, and Fig. 5c presents results consistent with the conclusion: A suitable α value ensures that the global gradient remains closer to its true direction, which leads to more consistent and effective updates during the learning process. This consistency improves the optimization trajectory, enabling the model to achieve higher accuracy compared to other state-of-the-art algorithms. Consequently, to mitigate the issues of non-convergence and oscillations observed respectively in Fig. 3, an α value of 0.5 is posited as the optimal threshold. This threshold is instrumental in characterizing the overall local gradient by incorporating a selectively proportioned segment, thereby ensuring the algorithm's effective performance.

4) Visualization demonstrating the validity of the gradients selected by BHerd: In order to more visually see the effect that the proposed algorithm produces on $\boldsymbol{g}_{(t,\sigma_i^\pi)}$, we show the results on five clients for **Case 2**, and the rest of the **Case 1** is in Appendix C.

As shown in Fig. 6, most of the $\boldsymbol{g}_{(t,\sigma_i^\pi)}$ on each client selected by the BHerd strategy is concentrated in a fixed domain. This means that in the FL system, regardless of training IID or Non-IID local datasets, gradients with large deviations from the local average gradient $\boldsymbol{\mu}_{(t,\sigma_i)}$ always exist on each client in each round. Therefore, the BHerd strategy is to exclude these deviating gradients to select the remaining beneficial gradients, and this 'benefit' is reflected in accelerating the convergence of the global model to the optimal. Meanwhile, it explains the meaning of α: the proportion of beneficial gradients in the local gradient.

In Proposition 1, we replace $\boldsymbol{\mu}_{(t,\sigma_i)}$ with $\frac{1}{\alpha\tau_i}\boldsymbol{g}_{(t,\sigma_i^\pi)}$ due to the consistency of the local objective. Thus, in Fig. 6, it can be obtained that the distance values of all the clients in the corresponding **Case 3** are in a small range and decrease with the increase of epoch. For **Case 3**, Client 1-3 are assigned half of the total dataset according to **Case 1** (IID) and thus have similar curves and the distance values remain minimal. While Client 4-5 are assigned the other half of the total dataset according to **Case 2** (Non-IID) and thus have different curves and larger distance values. These experimental results show that the BHerd strategy coincides with Proposition 1 and is

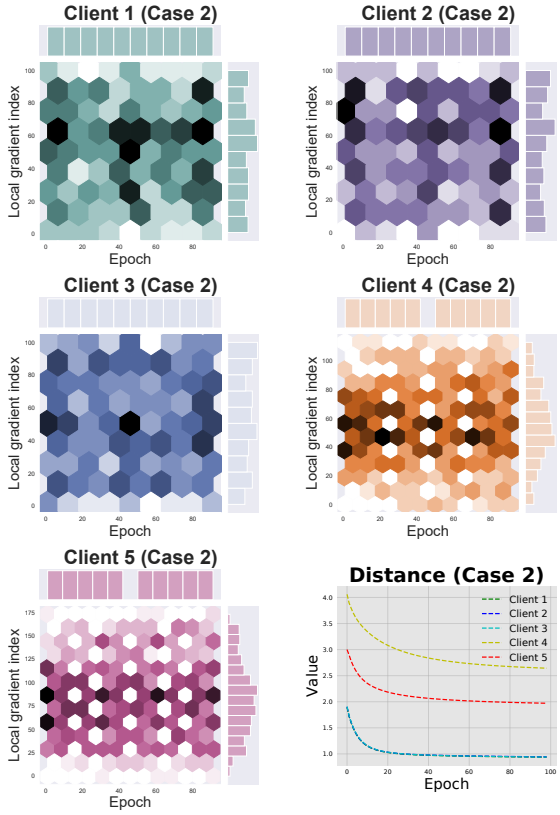


Fig. 6. Distribution of gradients selected by BHerd on each epoch for five clients in **Case 2**, and the distance between $\frac{1}{\alpha\tau_i}\boldsymbol{g}_{(t,\sigma_i^\pi)}$ and $\boldsymbol{\mu}_{(t,\sigma_i)}$.

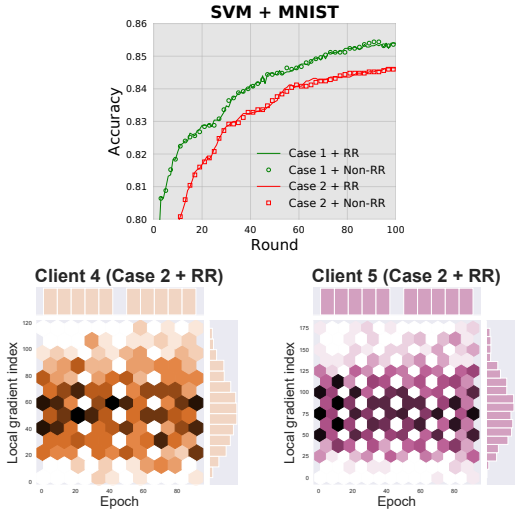


Fig. 7. Classification accuracy values of BHerd (SVM + MNIST) in **Case 1-2** with the RR and Non-RR protocol, and the distribution of gradients selected for Client 4 and 5 in **Case 2** with the RR protocol.

effective.

All the above analysis is based on the fact that the proposed algorithm does not use the RR protocol, so we analyze the impact of RR in the next section.

5) Impact of Random Reshuffling Protocol on System Performance: In order to validate our strategy of not using the RR protocol as mentioned in Section III-A, we do a comparative

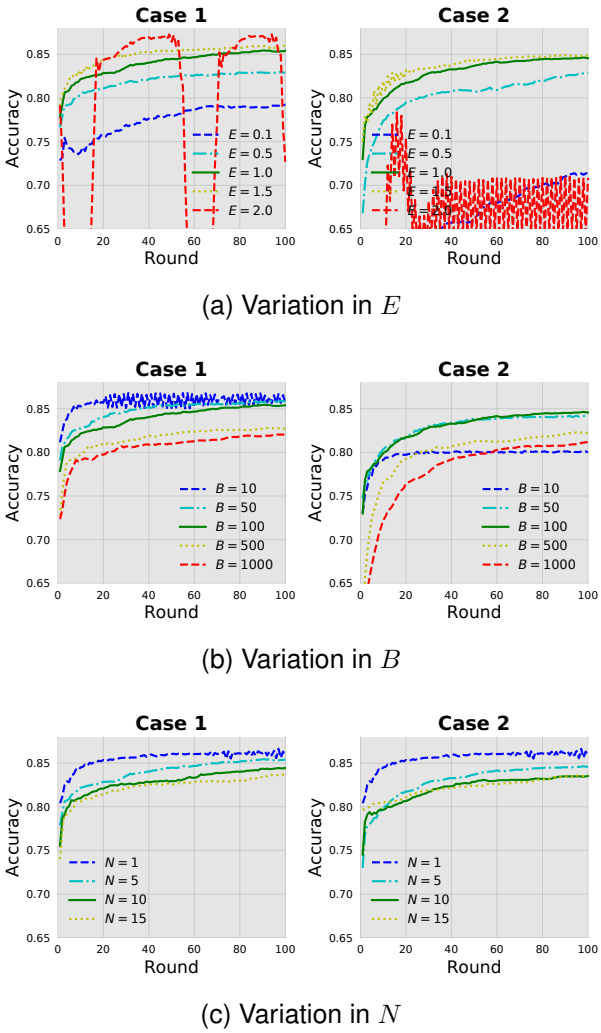


Fig. 8. Classification accuracy values (SVM + MNIST) in **Case 1-2** with the different values of E, B and N.

experiment on the RR protocol based on the experimental setup ($\alpha = 0.5$) in Section V-B3. The results of the accuracy are shown in Fig. 7, we can see that in **Case 1-2**, whether or not using the RR protocol has little impact on the convergence curves and the final results. Furthermore, in Fig. 7, for Client 4 and 5 with Non-IID data, RR does not affect the gradient selection in BHerd.

In the next experiments, we will investigate the effect of other parameters in the FL system on the BHerd algorithm.

6) Variation in Training Parameters: On a global level, the BHerd algorithm does a scaling transformation: $\tau_i \rightarrow \alpha\tau_i$. And according to the computation on $\tau_{(t,i)} = \lfloor E\frac{|\mathcal{D}_i|}{B} \rfloor$ in Section V-A4, the parameters that have an effect on τ_i are E, $|\mathcal{D}_i|$ (which is directly determined by N) and B. Therefore, we do ablation experiments on the three parameters E, B and N, respectively.

Fixing B and N constant, we only show the results of running 'SVM + MNIST' and using different values of E (0.1, 0.5, 1.0, 1.5 and 2.0) for the five clients in **Case 1-2**, The results of the accuracy are shown in Fig. 8a. It is observed that as E ascends from 0.1 to 2, there is an acceleration in the convergence of the depicted curves. This acceleration can be attributed to the increase in 'beneficial' gradients at each client, thereby facilitating a more rapid convergence of the model. Nevertheless, at $E = 2$, the model's convergence trajectory exhibits oscillations across rounds. This phenomenon can be attributed to the accumulation of local gradients, divergent from the global objective, resultant from excess local training steps. Such divergence amplifies the likelihood of our BHerd strategy selecting these aberrant gradients in certain rounds, consequently impeding. Furthermore, it is observed that the computational complexity of BHerd escalates exponentially with an increase in E, thereby necessitating the selection of $E = 1$ as a pragmatic value to balance efficiency and performance.

Fixing E and N constant, we only show the results of running 'SVM + MNIST' and using different values of B (10, 50, 100, 500 and 1000) for the five clients in **Case 1-2**, The results of the accuracy are shown in Fig. 8b. While E modulates the quantity of selected gradients by adjusting the aggregate gradient count per round, B, in contrast, influences the total count of local gradients, denoted as $\frac{|\mathcal{D}_i|}{B}$, given fixed E and N (i.e., $|\mathcal{D}_i|$). Owing to the unique operational dynamics of the BHerd strategy, the selection of B critically alters the distribution of local training datasets and, consequently, the number of local training gradients. This adjustment necessitates an optimal B value, which varies distinctly across different **Case** scenarios.

As delineated in Fig. 8b, the identified optimal B values for **Case 1-2** are 10 and 50, respectively. It is observed that above these optimal values, an increment in batch size inversely affects the model's convergence rate and predictive accuracy. A conceivable rationale for this phenomenon posits that lower B values enhance the likelihood of selecting more beneficial training features within local datasets through the herding operation, facilitating convergence to the local optimum on individual clients. Consequently, in scenarios where the training dataset is IID—thereby equating local optimization with global optimization—a reduced B value can expedite convergence towards the global optimum, maintaining a minimal distance value as discussed in Section V-B4. Conversely, in the presence of non-IID training data, it becomes imperative to adjust the B value upwards to a level that sustains a low distance value, thereby augmenting the dataset's generalization capability and averting convergence to suboptimal local minima.

Fixing E and B constant, we record the accuracy values of running 'SVM + MNIST' in **Case 1-3**, where the number of clients is chosen to be 1, 5, 10 and 15 for the proposed algorithm, and the results are shown in Fig. 8c. When $N = 1$, the BHerd algorithm is equivalent to centralized gradient descent, and thus its curves have the fastest convergence rate in **Case 1-2**, and next we analyze the impact of N from 5 to 15.

Unlike E, which affects the number of beneficial gradients selected, and B, which affects the selection rate of beneficial gradients, N affects the distribution of beneficial gradients by varying the size of $|\mathcal{D}_i|$. When N is small and the data is more centralized, it is easier for the FL model to quickly converge

to the local optimum for each client. This also explains that in **Case 1-3**, the curves with the fastest convergence and the best convergence results are $N = 5$.

Considering the overall experiments, we can conclude that the BHerd algorithm provides an efficient performance for FL, and provides a novel innovation with theory for FL optimization. In our future work, we aim to deploy the BHerd strategy efficiently across realistic edge environments and heterogeneous computing platforms. Furthermore, we will explore more sophisticated resource scheduling algorithms and present detailed empirical evaluations, including GPU scheduling mechanisms and communication overhead assessments.

VI. Conclusion and The Future Work

To accelerate the convergence of FL on Non-IID datasets, this paper proposes a BHerd algorithm to improve the generalization performance of the global model by ordering and pruning the local gradients, so as to reduce the detrimental effect of Non-IID datasets on the convergence of the global model. We prove the convergence of BHerd theoretically and demonstrate its effectiveness and superiority through extensive experiments. In future work, we will reduce the amount of computation and storage in BHerd, which is important for deploying large models and applying it to real-world scenarios. We will also refine the theoretical part of the application of BHerd on different frameworks. And further experiments will be conducted on more complex Non-IID scenarios, such as uneven datasize distributions and different Dirichlet distributions.

Acknowledgments

This work is sponsored in part by the National Natural Science Foundation of China under Grant No. 62025208, 62421002 and 62376278, Young Elite Scientists Sponsorship Program by CAST (No.2022QNRC001).

References

[1] S. S. Gill, H. Wu, P. Patros, C. Ottaviani, P. Arora, V. C. Pujol, D. Haunschild, A. K. Parlikad, O. Cetinkaya, H. Lutfiyya *et al.*, "Modern computing: Vision and challenges," *Telematics and Informatics Reports*, p. 100116, 2024.

[2] M. Phothilimthana, S. Abu-El-Haija, K. Cao, B. Fatemi, M. Burrows, C. Mendis, and B. Perozzi, "Tpugraphs: A performance prediction dataset on large tensor computational graphs," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[3] Z. Ning, H. Hu, X. Wang, L. Guo, S. Guo, G. Wang, and X. Gao, "Mobile edge computing and machine learning in the internet of unmanned aerial vehicles: a survey," *ACM Computing Surveys*, vol. 56, no. 1, pp. 1–31, 2023.

[4] S. Iftikhar, S. S. Gill, C. Song, M. Xu, M. S. Aslanpour, A. N. Toosi, J. Du, H. Wu, S. Ghosh, D. Chowdhury *et al.*, "Ai-based fog and edge computing: A systematic review, taxonomy and future directions," *Internet of Things*, vol. 21, p. 100674, 2023.

[5] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan, "Adaptive federated learning in resource constrained edge computing systems," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1205–1221, 2019.

[6] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *arXiv preprint arXiv:1610.05492*, 2016.

[7] S. Guo, K. Zhang, B. Gong, L. Chen, Y. Ren, F. Qi, and X. Qiu, "Sandbox computing: A data privacy trusted sharing paradigm via blockchain and federated learning," *IEEE Transactions on Computers*, vol. 72, no. 3, pp. 800–810, 2022.

[8] Y. Liu, S. Guo, J. Zhang, Z. Hong, Y. Zhan, and Q. Zhou, "Collaborative neural architecture search for personalized federated learning," *IEEE Transactions on Computers*, 2024.

[9] Y. Zhan, P. Li, L. Wu, and S. Guo, "L4l: Experience-driven computational resource control in federated learning," *IEEE Transactions on Computers*, vol. 71, no. 4, pp. 971–983, 2021.

[10] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*. PMLR, 2017, pp. 1273–1282.

[11] J. Wang and G. Joshi, "Cooperative sgd: A unified framework for the design and analysis of local-update sgd algorithms," *Journal of Machine Learning Research*, vol. 22, 2021.

[12] P. Dvurechensky, A. Gasnikov, and A. Kroshnin, "Computational optimal transport: Complexity by accelerated gradient descent is better than by sinkhorn's algorithm," in *International conference on machine learning*. PMLR, 2018, pp. 1367–1376.

[13] Y. Lei, T. Hu, G. Li, and K. Tang, "Stochastic gradient descent for nonconvex learning without bounded gradient assumptions," *IEEE transactions on neural networks and learning systems*, vol. 31, no. 10, pp. 4394–4400, 2019.

[14] N. J. Harvey, C. Liaw, Y. Plan, and S. Randhawa, "Tight analyses for non-smooth stochastic gradient descent," in *Conference on Learning Theory*. PMLR, 2019, pp. 1579–1613.

[15] D. Li, Z. Lai, K. Ge, Y. Zhang, Z. Zhang, Q. Wang, and H. Wang, "Hpdl: towards a general framework for high-performance distributed deep learning," in *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2019, pp. 1742–1753.

[16] T. Sun, B. Wu, D. Li, and K. Yuan, "Gradient normalization makes heterogeneous distributed learning as easy as homogenous distributed learning," 2022.

[17] T. Sun, D. Li, and B. Wang, "Decentralized federated averaging," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 4, pp. 4289–4301, 2022.

[18] M. Aledhari, R. Razzak, R. M. Parizi, and F. Saeed, "Federated learning: A survey on enabling technologies, protocols, and applications," *IEEE Access*, vol. 8, pp. 140 699–140 725, 2020.

[19] F. Sattler, S. Wiedemann, K.-R. Müller, and W. Samek, "Robust and communication-efficient federated learning from non-iid data," *IEEE transactions on neural networks and learning systems*, vol. 31, no. 9, pp. 3400–3413, 2019.

[20] M. Luo, F. Chen, D. Hu, Y. Zhang, J. Liang, and J. Feng, "No fear of heterogeneity: Classifier calibration for federated learning with non-iid data," *Advances in Neural Information Processing Systems*, vol. 34, pp. 5972–5984, 2021.

[21] P. Zhang, H. Sun, J. Situ, C. Jiang, and D. Xie, "Federated transfer learning for iiot devices with low computing power based on blockchain and edge computing," *Ieee Access*, vol. 9, pp. 98 630–98 638, 2021.

[22] M. Paul, S. Ganguli, and G. K. Dziugaite, "Deep learning on a data diet: Finding important examples early in training," *Advances in Neural Information Processing Systems*, vol. 34, pp. 20 596–20 607, 2021.

[23] S. Yang, Z. Xie, H. Peng, M. Xu, M. Sun, and P. Li, "Dataset pruning: Reducing training data by examining generalization influence," *arXiv preprint arXiv:2205.09329*, 2022.

[24] Y. Lu, W. Guo, and C. M. De Sa, "Grab: Finding provably better data permutations than random reshuffling," *Advances in Neural Information Processing Systems*, vol. 35, pp. 8969–8981, 2022.

[25] J. Cheng, P. Luo, N. Xiong, and J. Wu, "Aafl: Asynchronous-adaptive federated learning in edge-based wireless communication systems for countering communicable infectious diseasess," *IEEE Journal on Selected Areas in Communications*, vol. 40, no. 11, pp. 3172–3190, 2022.

[26] A. Fallah, A. Mokhtari, and A. Ozdaglar, "Personalized federated learning with theoretical guarantees: A model-agnostic meta-learning approach," *Advances in Neural Information Processing Systems*, vol. 33, pp. 3557–3568, 2020.

[27] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *International conference on machine learning*. PMLR, 2017, pp. 1126–1135.

[28] C. T Dinh, N. Tran, and J. Nguyen, "Personalized federated learning with moreau envelopes," *Advances in Neural Information Processing Systems*, vol. 33, pp. 21 394–21 405, 2020.

[29] V. Smith, C.-K. Chiang, M. Sanjabi, and A. S. Talwalkar, "Federated multi-task learning," *Advances in neural information processing systems*, vol. 30, 2017.

[30] L. Corinzia, A. Beuret, and J. M. Buhmann, "Variational federated multi-task learning," *arXiv preprint arXiv:1906.06268*, 2019.

[31] F. Chen, M. Luo, Z. Dong, Z. Li, and X. He, "Federated meta-learning with fast convergence and efficient communication," *arXiv preprint arXiv:1802.07876*, 2018.

[32] Z. Zhu, J. Hong, and J. Zhou, "Data-free knowledge distillation for heterogeneous federated learning," in *International Conference on Machine Learning*. PMLR, 2021, pp. 12 878–12 889.

[33] T. Lin, L. Kong, S. U. Stich, and M. Jaggi, "Ensemble distillation for robust model fusion in federated learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 2351–2363, 2020.

[34] X. Peng, Z. Huang, Y. Zhu, and K. Saenko, "Federated adversarial domain adaptation," *arXiv preprint arXiv:1911.02054*, 2019.

[35] H. Zhu, J. Xu, S. Liu, and Y. Jin, "Federated learning on non-iid data: A survey," *Neurocomputing*, vol. 465, pp. 371–390, 2021.

[36] T. Yoon, S. Shin, S. J. Hwang, and E. Yang, "Fedmix: Approximation of mixup under mean augmented federated learning," in *International Conference on Learning Representations*, 2020.

[37] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," in *International Conference on Learning Representations*, 2018.

[38] K. Zhou, Z. Liu, Y. Qiao, T. Xiang, and C. C. Loy, "Domain generalization: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[39] A. Back de Luca, G. Zhang, X. Chen, and Y. Yu, "Mitigating data heterogeneity in federated learning with data augmentation," *arXiv e-prints*, pp. arXiv–2206, 2022.

[40] H. Wang, Z. Kaplan, D. Niu, and B. Li, "Optimizing federated learning on non-iid data with reinforcement learning," in *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*. IEEE, 2020, pp. 1698–1707.

[41] M. Yang, X. Wang, H. Zhu, H. Wang, and H. Qian, "Federated learning with class imbalance reduction," in *2021 29th European Signal Processing Conference (EUSIPCO)*. IEEE, 2021, pp. 2174–2178.

[42] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-iid data," *arXiv preprint arXiv:1806.00582*, 2018.

[43] T. Tuor, S. Wang, B. J. Ko, C. Liu, and K. K. Leung, "Overcoming noisy and irrelevant data in federated learning," in *2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE, 2021, pp. 5020–5027.

[44] N. Yoshida, T. Nishio, M. Morikura, K. Yamamoto, and R. Yonetani, "Hybrid-fl: Cooperative learning mechanism using non-iid data in wireless networks," *arXiv preprint arXiv:1905.07210*, 2019.

[45] K. Mishchenko, A. Khaled, and P. Richtárik, "Random reshuffling: Simple analysis with vast improvements," *Advances in Neural Information Processing Systems*, vol. 33, pp. 17 309–17 320, 2020.

[46] S. Horváth, M. Sanjabi, L. Xiao, P. Richtárik, and M. Rabbat, "Fedshuffle: Recipes for better use of local work in federated learning," *Transactions on Machine Learning Research*, 2022. [Online]. Available: https://openreview.net/forum?id=Lgs5pQ1v30

[47] J. Wang, Q. Liu, H. Liang, G. Joshi, and H. V. Poor, "Tackling the objective inconsistency problem in heterogeneous federated optimization," *Advances in neural information processing systems*, vol. 33, pp. 7611–7623, 2020.

[48] S. P. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, and A. T. Suresh, "Scaffold: Stochastic controlled averaging for federated learning," in *International Conference on Machine Learning*. PMLR, 2020, pp. 5132–5143.

[49] A. E. Durmus, Z. Yue, M. Ramon, M. Matthew, W. Paul, and S. Venkatesh, "Federated learning based on dynamic regularization," in *International conference on learning representations*, 2021.

[50] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[51] A. Krizhevsky *et al.*, "Learning multiple layers of features from tiny images," 2009.

[52] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning word vectors for sentiment analysis," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Portland, Oregon, USA: Association for Computational Linguistics, June 2011, pp. 142–150. [Online]. Available: http://www.aclweb.org/anthology/P11-1015

[53] L. N. Smith, "Cyclical learning rates for training neural networks," in *2017 IEEE winter conference on applications of computer vision (WACV)*. IEEE, 2017, pp. 464–472.

[54] L. Bottou, F. E. Curtis, and J. Nocedal, "Optimization methods for large-scale machine learning," *SIAM review*, vol. 60, no. 2, pp. 223–311, 2018.

Ping Luo is currently pursuing the Ph.D. degree in Computer Science and Technology from National University of Defense Technology (NUDT), Changsha, China. He received his MA.Eng. degree in Computer Science and Technology with Hainan University, Haikou, China. His research interests include Convex Optimization, the Industrial Internet of Things and Artificial Intelligence.

Xiaoge Deng received the B.S. degree in Mathematics from the University of Science and Technology of China (USTC), Hefei, China, in 2018. He obtained the M.S. and Ph.D. degrees in Computer Science from the College of Computer Science and Technology at the National University of Defense Technology (NUDT), Changsha, China, in 2020 and 2024, respectively. Currently, he is an Assistant Professor at the Intelligent Game and Decision Lab in Beijing, China. His research interests include optimization algorithms and distributed machine learning.

Ziqing Wen received the BS degree in mathematics from SouthWest Jiao Tong University (SWJTU), Chengdu, China in 2022. He is currently pursuing the Phd degree at the School of Computer National University of Defense Technology (NUDT). His research interests include optimization for machine learning, generative AI.

Tao Sun received the BS, MS, and PhD degrees in mathematics from the National University of Defense Technology, Changsha, China in 2012, 2015, and 2018, respectively. He is currently an assistant professor at the National Laboratory for Parallel and Distributed Processing, National University of Defense Technology. His research interests include optimization for machine learning, image processing, distributed systems, and neural networks.

Dongsheng Li received the BSc (Hons.) and PhD (Hons.) degrees in computer science from the National University of Defense Technology, Changsha, China in 1999 and 2005, respectively. He is currently a full professor at the National Laboratory for Parallel and Distributed Processing, National University of Defense Technology. His research interests include parallel and distributed computing, cloud computing, and large-scale data management. He was awarded the prize of the National Excellent Doctoral Dissertation of China by the Ministry of Education of China in 2008. His research interests include distributed systems, cloud computing, and Big Data processing.