FedDIP: Federated Learning with Extreme Dynamic Pruning and Incremental Regularization

Qianyu Long
School of Computing Science
University of Glasgow, UK
2614994L@student.gla.ac.uk

Christos Anagnostopoulos
School of Computing Science
University of Glasgow, UK
christos.anagnostopoulos@glasgow.ac.uk

Shameem Puthiya
School of Computing Science
University of Glasgow, UK
Sham.Puthiya@glasgow.ac.uk

Daning Bi
Hunan University, China
College of Finance and Statistics
daningbi@hnu.edu.cn

***Abstract*—Federated Learning (FL) has been successfully adopted for distributed training and inference of large-scale Deep Neural Networks (DNNs). However, DNNs are characterized by an extremely large number of parameters, thus, yielding significant challenges in exchanging these parameters among distributed nodes and managing the memory. Although recent DNN compression methods (e.g., sparsification, pruning) tackle such challenges, they do not holistically consider an adaptively controlled reduction of parameter exchange while maintaining high accuracy levels. We, therefore, contribute with a novel FL framework (coined FedDIP), which combines (i) dynamic model pruning with error feedback to eliminate redundant information exchange, which contributes to significant performance improvement, with (ii) incremental regularization that can achieve *extreme* sparsity of models. We provide convergence analysis of FedDIP and report on a comprehensive performance and comparative assessment against state-of-the-art methods using benchmark data sets and DNN models. Our results showcase that FedDIP not only controls the model sparsity but efficiently achieves similar or better performance compared to other model pruning methods adopting incremental regularization during distributed model training.**

***Index Terms*—Federated Learning, dynamic pruning, extreme sparsification, incremental regularization.**

I. Introduction

Federated Learning (FL) [1] is a prevalent *distributed learning* paradigm due to its ability to tackle learning at scale. FL plays a significant role in large-scale predictive analytics by enabling the decentralisation of knowledge discovery. FL contributes towards privacy preservation, which overcomes fundamental issues of data governance and ownership [2]. Distributed training and deploying large-scale Machine Learning (ML) models, i.e., Deep Neural Networks (DNNs), impose significant challenges due to the huge volumes of training data, *large* models, and diversity in data distributions.

Distributed computing nodes, mainly located at the network edge being as close to data sources as possible, collaboratively engineer ML models rather than depending on collecting all the data to a centralized location (data centre or Cloud) for training [3]. This computing paradigm, coined Edge Computing, has been successfully applied to various predictive modelling, mining, and analytics applications, e.g., in finance [4], healthcare [5] and wireless sensor networks [6].

DNNs are characterised by an extremely large number of parameters. For instance, the Convolutional Neural Networks (CNN) ResNet50 [7] and VGG16 [8] consist of 27 and 140 million parameters, respectively, while generative AI models, like GPT-2 have more than 1.5 billion parameters [9]. Evidently, this places a great burden on distributed computing nodes when exchanging model parameters during training, tuning, and inference.

Model size reduction (pruning) methods, e.g., [10], [11], [12] aim to retain the prediction accuracy while reducing the communication overhead by decreasing the number of model parameters exchanged among nodes. However, most pruning methods focus on the compression of model gradients. Even though they yield high compression rates, they do not achieve significantly compact models for exchange. But in general, methods that can produce compact models along with significant redundancy in the number of DNN weights by sophisticatedly pruning the weights are deemed appropriate [13]. In contrast to model gradient compression, model weight compression significantly shrinks the model size by setting most of the weights to zero. This is desirable for eliminating redundancy in model exchange during distributed knowledge extraction. But often such models result in performance degradation. Therefore, the question we are addressing is: *How to effectively introduce model pruning mechanisms in a decentralized learning setting that is capable of achieving extremely high compression rates while preserving optimal predictive performance?* We contribute with an efficient method based on *dynamic pruning with error feedback* and *incremental regularization*, coined FedDIP. FedDIP's novelty lies in the principle of adapting dynamic pruning in a decentralized way by pushing unimportant weights to zeros (extreme pruning) whilst maintaining high accuracy through incremental regularization. To the best of our knowledge, FedDIP is the first approach that combines incremental regularization and extreme dynamic pruning in FL.

The paper is organized as follows: Section II reports on related work and our contribution. Section III provides preliminaries in FL and model pruning methods. Section IV elaborates on the FedDIP framework, while Section V reports on the theoretical properties of FedDIP and convergence analysis. Our experimental results in Section VI showcase the efficiency of FedDIP in distributed learning. Section VII concludes the paper with future research directions.

II. Related Work & Contribution

A. Model Gradient & Model Weight Sparsification

Expensive and redundant sharing of model weights is a significant obstacle in distributed learning [14]. The size of the exchanged models among nodes can be reduced by compression and sparsification. The work in [11] adopts magnitude selection on model gradients to yield sparsification when using Stochastic Gradient Descent (SGD). Instead of dense updates of weights, [10] proposed a distributed SGD that keeps 1% of the gradients by comparing their magnitude values. The method in [15] scales up SGD training of DNN via controlling the rate of weight update per individual weight. [16] develops encoding SGD-based vectors achieving reduced communication overhead. [17] proposed the periodic quantized averaging SGD strategy that attains similar model predictive performance while the size of shared model gradients is reduced 95%. In [18], the authors argued that 99% of gradients are redundant and introduced a deep gradient compression method, which achieves compression rates in the range 270-600 with sacrificing accuracy. The *gTop-k* gradient sparsification method in [19] reduces communication cost based on the *Top-k* method in [18]. [20] develops a method based on [21] that adaptively compresses the size of exchanged model gradients via quantization.

In contrast to gradient sparsification, the shrinkage of the *entire* model size is of paramount importance in distributed learning. It not only eliminates communication redundancy during training but also enables less storage and inference time, which makes FL welcome in distributed knowledge systems. However, so far, only centralized learning adopts model compression via, e.g., weight pruning, quantization, low-rank factorization, transferred convolutional filters, and knowledge distillation [22], with pruning being our focus in this work. SNIP [23] introduces a method that prunes a DNN model once (i.e., prior to training) based on the identification of important connections in the model. [24] proposes a centralized two-step method that prunes each layer of a DNN via regression-based channel selection and least squares reconstruction. The method in [25] prunes CNNs centrally using the Alternating Direction Method of Multipliers (ADMM). Following [25], the PruneTrain method [26] uses structured group-LASSO regularization to accelerate CNN training in a centralised location only. The DPF [27] method allows dynamic management of the model sparsity with a feedback mechanism that re-activates pruned weights.

B. Contribution

Most of the approaches in FL take into account only the communication overhead and thus adopt gradient sparsification. Nonetheless, weight sparsification is also equally important and can lead to *accurate* distributed sparse models. Such sparse models are lightweight, thus, suitable for storage, transfer, training, and fast inference. As shown in [28], model weights and gradients averaging policies are *equivalent* only when the local number of model training epochs equals one. FedDIP tries to bridge the gap of weights average pruning in FL by obtaining highly accurate sparse models through incremental regularization and reducing communication during training through dynamic pruning.

To the best of our knowledge in distributed learning, PruneFL [12] and LotteryFL [29] methods attempt model pruning. However, LotteryFL focuses on a completely different problem from ours. LotteryFL tries to *discover sparse local* sub-networks (a.k.a. Lottery Ticket Networks) of a base DNN model. In contrast, FedDIP searches for a *sparse global* DNN model with mask readjustments on a central server, as we will elaborate on later. PruneFL starts with a pre-selected node to train a global shared mask function, while FedDIP generates the mask function with weights following the Erdős-Renéyi-Kernel (ERK) distribution [30], as we will discuss in the later sections. Our technical contributions are:

- An innovative federated learning paradigm, coined FedDIP, that combines extreme sparsity-driven model pruning with incremental regularization.
- FedDIP achieves negligible overhead keeping accuracy at the same or even higher levels over extremely pruned models.
- Theoretical convergence and analysis of FedDIP.
- A comprehensive performance evaluation and comparative assessment of FedDIP with benchmark i.i.d. and non-i.i.d. datasets and DNN models. Our experimental results reveal that FedDIP, in the context of high model compression rates, delivers superior prediction performance compared to the baseline methods and other approaches found in the literature, specifically, FedAvg [1], PruneFL [12], PruneTrain [26], DPF [27], and SNIP [23].

Notations	Definition
N, K	N: total number of nodes, where $K < N$ nodes participated in each training round
n indexes a node; z indexes a DNN layer; $n \in [N], z \in [Z]$ $[N]$ abbreviates the integer sequence $1, 2, \ldots, N$	
$\mathcal{D}_n, D_n$	Dataset and its size on node n.
$(\mathbf{x}, y) \in \mathcal{D}_n$	$\mathbf{x}, y$ are features and labels in node n's dataset
$f(\cdot), \nabla f(\cdot)$	Loss function and its derivative
ρ_n, η	Weight percentage and learning rate
$\boldsymbol{\omega}_G$, $\boldsymbol{\omega}_n$, $\boldsymbol{\omega}'_n$	Global, local and pruned local model parameters
T, L, τ, ℓ	Global and local rounds, global and local epochs
λ	Regularization hyperparameter
$\odot$	Element-wise (Hadamard) product
s_0, s_t, s_p	initial sparsity, sparsity at round t, final sparsity

TABLE I: Table of Notations

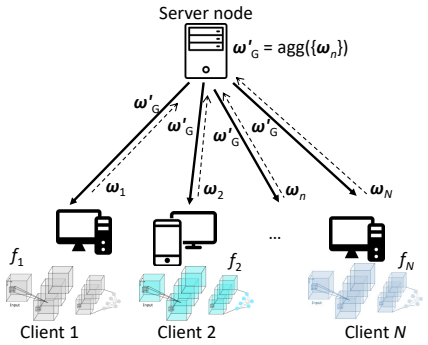


Fig. 1: An instance of the FedDIP framework.

III. Preliminaries

A. Federated Learning

For the general notations and definitions, please refer to Table I. Consider a distributed learning system involving a set of N nodes (clients) $\mathcal{N} = \{1, 2, \ldots, N\}$. Let $\mathcal{D}_n = \{(\mathbf{x}, y)\}$ be the local dataset associated with a node $n \in \mathcal{N}$ such that $\mathbf{x} \in \mathcal{X} \subset \mathbb{R}^d$, $y \in \mathcal{Y} \subset \mathbb{R}$, and $D_n = |\mathcal{D}_n|$. In the standard FL setting, given a subset of $K < N$ nodes $\mathcal{N}_c \subset \mathcal{N}$, the local loss is given by:

$$f_n(\boldsymbol{\omega}) = \frac{1}{D_n} \sum_{(\mathbf{x},y)\in\mathcal{D}_n} \mathcal{L}(\mathcal{G}(\boldsymbol{\omega}, \mathbf{x}), y) \tag{1}$$

where $\boldsymbol{\omega}$ is the model parameter, $\mathcal{G}$ is the discriminant function that maps the input space to output space and $\mathcal{L}$ is a loss function that measures the quality of the prediction , e.g., mean-squared-error, maximum likelihood, cross-entropy loss. The global loss function for all the selected nodes $n \in \mathcal{N}_c$ is:

$$f(\boldsymbol{\omega}) = \sum_{n\in\mathcal{N}_c} \rho_n f_n(\boldsymbol{\omega}), \text{ where } \rho_n = \frac{D_n}{\sum_{j\in\mathcal{N}_c} D_j}. \tag{2}$$

The model training process spans periodically over T global rounds with L local rounds. Let $t \in= \{0, 1, \ldots, T-1\}$ be a discrete-time instance during the training process. Then, $\tau = \lfloor \frac{t}{L} \rfloor L$ is the start time of the current global epoch. At τ, the nodes (clients) receive updated aggregated weights $\bar{\boldsymbol{\omega}}^\tau$ from the node responsible for aggregating the nodes' model parameters, a.k.a. the server node. The local training at client n at local epoch $l = 1, \ldots, L$ proceeds as:

$$\boldsymbol{\omega}_n^{(\tau+l)+1} = \boldsymbol{\omega}_n^{\tau+l} - \eta_{\tau+l} \nabla f_n(\boldsymbol{\omega}_n^{\tau+l}), \tag{3}$$

where $\eta \in (0, 1)$ is the learning rate. The weight averaging policy on the server node can be written as:

$$\bar{\boldsymbol{\omega}}^\tau \quad = \quad \sum_{n\in\mathcal{N}} \rho_n \boldsymbol{\omega}_n^\tau. \tag{4}$$

B. Model Pruning

In centralized learning systems (e.g., in Cloud), where all data are centrally stored and available, the model pruning [31] aims to sparsify various connection matrices that represent the weights of the DNN models. Notably, *sparsity*, hereinafter noted by $s \in [0, 1]$, indicates the proportion of non-zero weights among overall weights. A 100% sparse ($s = 1$) model indicates that all the weights are negligible (their values are close to 0), while a 0% sparse ($s = 0$) model stands for the full model with original weight values. Typically, the reduction of the number of nonzero weights (pruning) of a DNN model is achieved using *mask functions*. A mask function $\mathbf{m}$ acts like an indicator function that decides whether the parameter/weight at a certain position in a layer of a DNN model is zero or not. The model pruning based on mask functions requires a criterion to select the parameters to prune. The most common pruning criterion considers the absolute value of the weights of each parameter in a layer. Generally, a parameter is removed from the training process if its absolute value of the weight is less than a predefined threshold.

On the other hand, model pruning in FL is vital in light of reducing communication cost in *each* training round. Moreover, the global number of rounds should be reduced as this significantly contributes to the overall communication overhead. Hence, in FL, pruning aims at *extreme* model compression rates, i.e., $s \geq 0.8$ with a relatively small compromise in prediction accuracy. It is then deemed appropriate to introduce a distributed and adaptive pruning method with relatively high and controlled DNN model sparsity, which reduces communication costs per round along with ensuring convergence under high sparsity with only marginal decrease in prediction accuracy.

The pruning techniques are typically categorized into three: *pruning before training* (e.g., SNIP [23]), *pruning during training* (e.g., PruneTrain [26] and PruneFL [12]), and *pruning after training* DPF [27]. In this work, we concentrate on the two former techniques, which deal with efficient model training. The *pruning after training* approach offers limited utility in the context of distributed learning. The two commonly employed techniques for pruning are: (i) Regularization-based Pruning (RP) and (ii) Importance-based Pruning (IP) [32]. The interested reader may refer to [24], [25], [32] and the references therein for a comprehensive survey of RP and IP techniques. RP uses intrinsic sparsity-inducing properties of L_1 (Manhattan distance) and L_2 (Euclidean distance) norms to limit the *importance* of different model parameters. The sparsity-inducing norms constrain the weights of the unimportant parameters to small absolute values during training. Moreover, RP can effectively constrain the weights into a sparse model space via tuning the regularization hyperparameter λ. Whereas in IP, parameters are pruned purely based on predefined formulae that are defined in terms of the weights of the parameters or the sum of the weights. IP techniques are originally proposed in the unstructured pruning settings that can result in sparse models not capable of speeding up the computation. Even though RP techniques are considered superior to IP techniques, they struggle with two fundamental challenges: (**C1**) The first challenge pertains to controlling the sparsity value s during pruning. For example, in PruneTrain [26], employing a pruning threshold value of 10^{-4} to eliminate model parameters does not guarantee the delivery of a sparse model. (**C2**) The second challenge is dynamically tuning a reg-

ularization parameter λ. A large λ leads to model divergence during training, as the model may excessively lean towards penalty patterns. By adding regularization terms in DNN training traditionally aims for overfitting issues. However, additional regularization for prunable layers is required for RP, which is the core difference between traditional training and RP-based training.

IV. The FedDIP Framework

The proposed FedDIP framework integrates extreme dynamic pruning *with* error feedback and incremental regularization in distributed learning environments. Figure 1 illustrates a schematic representation of the FedDIP, which will be elaborated on in this section. FedDIP attempts to effectively train pruned DNN models across collaborative clients ensuring convergence by addressing the two challenges **C1** and **C2** prevalent in RP-based methods discussed in Section III-B.

The dynamic pruning method (DPF) in [27] demonstrates improved performance in comparison with other baselines under high sparsity. Given the SGD update scheme, the model gradient in DPF is computed on the pruned model as:

$$\boldsymbol{\omega}_{t+1} = \boldsymbol{\omega}_t - \eta_t \nabla f(\boldsymbol{\omega}'_t) = \boldsymbol{\omega}_t - \eta_t \nabla f(\boldsymbol{\omega}_t \odot \mathbf{m}_t), \quad (5)$$

taking into account the error feedback (analytically):

$$\boldsymbol{\omega}_{t+1} = \boldsymbol{\omega}_t - \eta_t \nabla f(\boldsymbol{\omega}_t + \mathbf{e}_t), \quad (6)$$

where $\mathbf{e}_t = \boldsymbol{\omega}'_t - \boldsymbol{\omega}_t$. In (5), $\odot$ represents the Hadamard (element-wise) product between the two model weights, $\boldsymbol{\omega}_t$ represents the entire model parameters, $\boldsymbol{\omega}'_t$ represents the pruned model parameters, and $\mathbf{m}$ is the adopted mask function used for pruning as in, e.g., in [12], [26], and [27]. The mask is applied on the model parameters $\boldsymbol{\omega}_t$ to eliminate weights according to the magnitude of each weight, thus, producing the pruned $\boldsymbol{\omega}'_t$. Applying the gradient, in this case, allows recovering from errors due to premature masking out of important weights, i.e., the rule in (5) takes a step that best suits the pruned model (our target). In contrast, all the pruning methods adopted in FL, e.g., [12], led to sub-optimal decisions by adopting the rule:

$$\boldsymbol{\omega}_{t+1} = \boldsymbol{\omega}'_t - \eta_t \nabla f(\boldsymbol{\omega}'_t). \quad (7)$$

One can observe that the update rule in (5) retains more information, as it only computes gradients of the pruned model, compared to the update rule in (7). This is expected to yield superior performance under high sparsity.

Moreover, it is known that the multi-collinearity[1] challenge is alleviated by the Least Absolute Shrinkage and Selection Operator (LASSO). LASSO performs simultaneous variable selection and regularisation [33]. LASSO adds the L_1 regularization term to the regression loss function, providing a solution to cases where the number of model parameters is significantly larger than the available observations. Apparently, this is the case in DNNs, which typically involve millions of parameters with only tens of thousands of observations. The two challenges reported in Section III-B deal with selecting appropriate dynamic policies for sparsity control and regularization hyperparameter λ. To address the challenge **C1**, we dynamically drop the least $s \cdot 100\%$ percentile according to weights magnitude. The challenge **C2** is addressed by incrementally increasing the regularization parameter departing from the principles of LASSO regression. It is also evidenced in [32] that growing regularization benefits pruning. Based on these observations, we establish the FedDIP algorithm to maintain the predictive model performance under extreme sparsity with incremental regularization and dynamic pruning. To clarify terminology, we refer to our algorithm that directly applies dynamic pruning as 'FedDP' (addressing challenge **C1**), while 'FedDIP' represents the variant that also adds incremental regularization (addressing both challenges **C1** and **C2**). Collectively, we refer to these variants as 'FedD(I)P'. Each node $n \in \mathcal{N}$ first trains a local sparse DNN model, which contains weights with relatively small magnitudes (see also Fig. 1). Then, the node n optimizes the proposed *local incrementally regularized loss function* at round t as:

$$f_n(\boldsymbol{\omega}_t) = \frac{1}{D_n} \sum_{(\mathbf{x},y)\in\mathcal{D}} \mathcal{L}(G(\boldsymbol{\omega}_t, \mathbf{x}), y) + \lambda_t \sum_{z=1}^{Z} \|\boldsymbol{\omega}_t^{(z)}\|_2, \quad (8)$$

where the step t dependent regularization parameter λ_t controls the degree of model shrinkage, i.e., the *sparsity*, and Z is the number of the DNN layers (this, of course, depends on the DNN architecture; in our experiments, it is the sum of convolutional and fully connected layers). The norm $\|\boldsymbol{\omega}^{(z)}\|_2 = (\sum_k |\omega_k^{(z)}|^2)^{1/2}$ is the L_2 norm of the pruned z^{th} layer of model weights $\boldsymbol{\omega}^{(z)}$. We then introduce the incremental regularization over λ_t based on the schedule:

$$\lambda_t = \begin{cases} 0 & \text{if } 0 \le t < \frac{T}{Q} \\ \vdots & \vdots \\ \frac{\lambda_{\max}\cdot(i-1)}{Q} & \text{if } \frac{(i-1)T}{Q} \le t < \frac{iT}{Q} \\ \vdots & \vdots \\ \frac{\lambda_{\max}(Q-1)}{Q} & \text{if } \frac{(Q-1)T}{Q} \le t \le T \end{cases} \quad (9)$$

with quantization step size $Q > 0$. The influence of Q on regularization is controlled by adapting λ_{max}. Such step size divides the regularization parameter space from $\frac{\lambda_{\max}}{Q}$ to $\lambda_{\max}$ to achieve a gradual increase of regularization at every $\frac{T}{Q}$ rounds. In addition, each node n adopts dynamic pruning to progressively update its local model weights $\boldsymbol{\omega}_n^{\tau+L}$ to optimize (8) as:

$$\boldsymbol{\omega}_n^{\tau,l+1} = \boldsymbol{\omega}_n^{\tau,l} - \eta_\tau \nabla f_n(\boldsymbol{\omega}_n'^{(\tau,l)}), \quad (10)$$

where $\boldsymbol{\omega}_n'^{(\tau+l)}$ is obtained through pruning based on a global mask function $\mathbf{m}_\tau$ generated by the server node. Moreover, our gradual pruning policy modifies the sparsity update policy per round from [34] by incrementally updating the sparsity as:

$$s_t = s_p + (s_0 - s_p)\Big(1 - \frac{t}{T}\Big)^3, \quad (11)$$

[1]In multi-collinearity, two or more independent variables are highly correlated in a regression model, which violates the *independence* assumption.

where s_t represents the sparsity applied to the model pruning at round t, s_0 is the initial sparsity, and s_p is the desired/target sparsity. Notably, in our approach s_0 is strictly non-zero; this can be a moderate sparsity of $s_0 = 0.5$. Such adaptation differentiates our method from [34], where $s_0 = 0$. In essence, we permit the sparsity to increment from moderate to extreme levels throughout the process. If considering $s_0 > 0$, the layer-wise sparsity of the initial mask follows the ERK distribution introduced in [30]. At the end of a local epoch l, the server node collects $K < N$ model weights $\boldsymbol{\omega}_n^{\tau+l}$ from the selected nodes $n \in \mathcal{N}_c$, and calculates the global weights average as:

$$\bar{\boldsymbol{\omega}}_G^{\tau+l} = \sum_{n \in \mathcal{N}} \rho_n \boldsymbol{\omega}_n^{\tau+l}. \tag{12}$$

In addition, the $\mathbf{m}_\tau$ mask function is generated based on pruning on $\bar{\boldsymbol{\omega}}_G^{\tau+l}$ with current sparsity s_τ. The FedDIP process is summarized in Algorithm 1, where *only* pruned models are exchanged from server to nodes, while pruning is *locally* achieved in the clients. **Note:** FedDIP achieves data-free initialization and generalizes the DPF [27] in dynamic pruning process. When we set initial $s_0 = 0$ and no incremental regularization, i.e., $\lambda_t = 0$, $\forall t$, then FedDIP reduces to DPF. Moreover, we obtain our variant FedDP if we set $\lambda_t = 0$, $\forall t$ with $s_0 > 0$ w.r.t. ERK distribution.

Remark 1. *FedDIP uses a reconfiguration horizon R during model training for updating the mask function. Specifically, for every R global round (when $\tau \mod R = 0$), the mask function $\mathbf{m}_\tau$ should be updated to obtain a smooth (accuracy) learning curve. Such a horizon is empirically derived. If the mask function is not updated during the horizon T, the model might converge to a local optimum. On the other hand, if the mask function is updated frequently, the updates in the model might not match the updates in the sparse model structure.*

Remark 2. *Since (8) might not be differentiable, we denote by $\nabla f_n(\boldsymbol{\omega}_n^{\tau+l})$ the gradients or sub-gradients of $f_n(\boldsymbol{\omega}_n^{\tau+l})$ and omit the explicit closed formula for the calculation of the gradient due to the various model architectures. In practice, this is solved by the auto-gradient function in [35].*

Algorithm 1 The FedDIP Algorithm

Input: N nodes; T global rounds; L local rounds; initial and target sparsity s_0 and s_p; maximum regularization $\lambda_{\max}$; quantization step Q; reconfiguration horizon R
Output: Global pruned DNN model weights $\boldsymbol{\omega}'_G$

//Server initiliazation
if $s_0 > 0$ **then**
  Server initializes global mask $\mathbf{m}_0$ (ERK distribution)
end if
//Node update & pruning
for global round $\tau = 1, \ldots, T$ **do**
  Server randomly selects K nodes $\mathcal{N}_c \subset \mathcal{N}$
  for selected node $n \in \mathcal{N}_c$ **in parallel do**
    Receive pruned weights $\boldsymbol{\omega}_G^{\prime(\tau-1)}$ from server node
    Obtain mask $\mathbf{m}_{\tau-1}$ from $\boldsymbol{\omega}_G^{\prime(\tau-1)}$
    Train $\boldsymbol{\omega}_n^\tau$ over L rounds on data $\mathcal{D}_n$ using (10)
    if incremental regularization is chosen **then**
      Optimize (8) with incremental λ_τ in (9)
    else
      Optimize (1)
    end if
  end for
  //Server update, aggregation & reconfiguration
  Server receives models and aggregates $\boldsymbol{\omega}_G^\tau$ in (12)
  if $\tau \mod R == 0$ **then**
    Reconfigure global mask $\mathbf{m}_\tau$ based on pruning $\boldsymbol{\omega}_G^\tau$
  end if
  Server prunes global model with $\mathbf{m}_\tau$ and obtains $\boldsymbol{\omega}_G^{\prime(\tau)}$
  Server node returns $\boldsymbol{\omega}_G^{\prime(\tau)}$ to all nodes.
end for

V. Theoretical & Convergence Analysis

In this section, we provide a theoretical analysis of FedDIP including the convergence Theorem 1 ensuring stability in training models w.r.t. incremental regularization and dynamic extreme pruning. **Note for Proofs:** *Due to space limitation, the proofs of our Theorem 1 and lemmas are omitted; the authors are committed to making them available on the accepted version of this paper.*

At each global round $t \in \{1, \ldots, T\}$, K out of N nodes participate, each one selected with probability ρ_n aligned with [36], [37] and $\sum_{n=1}^N \rho_n = 1$. Let $\boldsymbol{\omega}_n^t$ and $\boldsymbol{\omega}_n^{\prime(t)}$ be the weights and pruned ones at round t on node n, respectively, with

$$\boldsymbol{\omega}_n^{\prime(t)} = \boldsymbol{\omega}_n^t \odot \mathrm{m}^t. \tag{13}$$

Let also $\mathbf{v}_n^t$ and $\tilde{\mathbf{v}}_n^t$ be the expected and estimated gradients at t, respectively, on node n. Based on $\boldsymbol{\omega}_n^{\prime(t)}$, we obtain: $\mathbf{v}_n^{\prime(t)} = \nabla f(\boldsymbol{\omega}_n^{\prime(t)})$ while $\tilde{\mathbf{v}}_n^{\prime(t)}$ is the estimated one. The global aggregated model for FedAvg is:

$$\bar{\boldsymbol{\omega}}^t = \frac{1}{K} \sum_{n \in \mathcal{N}_c} \boldsymbol{\omega}_n^t, \tag{14}$$

while before the server sends the model, it is pruned as

$$\bar{\boldsymbol{\omega}}^{\prime(t)} = \frac{1}{K} \sum_{n \in \mathcal{N}_c} \boldsymbol{\omega}_n^t \odot \mathrm{m}^t. \tag{15}$$

The global estimated aggregated gradient and expected global gradient, respectively, are:

$$\tilde{\mathbf{v}}^t = \frac{1}{K} \sum_{n \in \mathcal{N}_c} \tilde{\mathbf{v}}_n^t \text{ and } \bar{\mathbf{v}}^t = \frac{1}{K} \sum_{n \in \mathcal{N}_c} \mathbf{v}_n^t. \tag{16}$$

Similarly, for DPF, we have that:

$$\tilde{\mathbf{v}}^{\prime(t)} = \frac{1}{K} \sum_{n \in \mathcal{N}_c} \tilde{\mathbf{v}}_n^{\prime(t)} \text{ and } \bar{\mathbf{v}}^{\prime(t)} = \frac{1}{K} \sum_{n \in \mathcal{N}_c} \mathbf{v}_n^{\prime(t)}. \tag{17}$$

In FedAvg, $\bar{\boldsymbol{\omega}}^t$ is updated as: $\bar{\boldsymbol{\omega}}^{t+1} = \bar{\boldsymbol{\omega}}^t - \eta_t \tilde{\mathbf{v}}^t$, while the update rule based on DPF at node n is:

$$\boldsymbol{\omega}_n^{t+1} = \boldsymbol{\omega}_n^t - \eta_t \tilde{\mathbf{v}}_n^{\prime(t)}, \tag{18}$$

where $\boldsymbol{\omega}_n^t = \bar{\boldsymbol{\omega}}'^{(t)}$. Similarly, $\bar{\boldsymbol{\omega}}^{t+1}$ is updated as:

$$\bar{\boldsymbol{\omega}}^{t+1} = \bar{\boldsymbol{\omega}}'^{(t)} - \eta_t \tilde{\mathbf{v}}'^{(t)}. \tag{19}$$

Definition 1. *According to [27], the quality of pruning is defined by the parameter $\delta_t \in [0,1]$ as:*

$$\delta_t := \frac{\|\boldsymbol{\omega}^t - \boldsymbol{\omega}'^{(t)}\|_F^2}{\|\boldsymbol{\omega}^t\|_F^2} \tag{20}$$

where $\|.\|_F^2$ is the square of Frobenius matrix norm. δ_t indicates the degree of information loss by pruning in terms of magnitude. A smaller δ_t stands for less information loss.

Definition 2. *Following the Definition 1 in [38], a measurement γ of non-i.i.d. (non-independent and identically distributed) data is defined as follows:*

$$\gamma = \frac{\sum_{n=1}^N p_n \|\nabla f_n(\boldsymbol{\omega})\|^2}{\|\sum_{n=1}^N p_n \nabla f_n(\boldsymbol{\omega})\|^2}, \tag{21}$$

with $\gamma \geq 1$; $\gamma = 1$ holds in i.i.d case.

We list our assumptions for proving the convergence of FedDIP in the learning phase.

Assumption 1. *$M-$**Smoothness**.* $, \forall \boldsymbol{\omega}^{t_1}, \boldsymbol{\omega}^{t_2} \in \mathbb{R}^d$, $M \in \mathbb{R}$

$$f(\boldsymbol{\omega}^{t_1}) \leq f(\boldsymbol{\omega}^{t_2}) + (\boldsymbol{\omega}^{t_1} - \boldsymbol{\omega}^{t_2})^\top \nabla f(\boldsymbol{\omega}^{t_2}) + \frac{M}{2}\|\boldsymbol{\omega}^{t_1} - \boldsymbol{\omega}^{t_2}\|^2$$

Assumption 2. *$\mu-$**Lipschitzness**.* $\forall \boldsymbol{\omega}^{t_1}, \boldsymbol{\omega}^{t_2} \in \mathbb{R}^d$ *and* $\mu \in \mathbb{R}$

$$\|f(\boldsymbol{\omega}^{t_1}) - f(\boldsymbol{\omega}^{t_2})\| \leq \mu \|\boldsymbol{\omega}^{t_1} - \boldsymbol{\omega}^{t_2}\| \tag{22}$$

Assumption 3. ***Bounded variance for gradients**. Following Assumption 3 in [37], the local model gradients on each node n are self-bounded in variance:*

$$\mathbb{E}[\|\tilde{\mathbf{v}}_n^t - \mathbf{v}_n^t\|^2] \leq \sigma_n^2. \tag{23}$$

Assumption 4. ***Bounded weighted aggregation of gradients.** Following Assumption 4 in [38], the aggregation of local gradients at time t are bounded as:*

$$\|\sum_{n=1}^N \rho_n \mathbf{v}_n^t\|^2 \leq G^2, \tag{24}$$

where $\sum_{n=1}^N \rho_n = 1$ and $\sum_{n=1}^N \rho_n \mathbf{v}_n^t$ stands for the weighted aggregation of local gradients; $G \in \mathbb{R}$.

Before providing the FedDIP convergence Theorem 1, we stress that:

$$\mathbb{E}[f(\bar{\boldsymbol{\omega}}'^{(t+1)}) - f(\bar{\boldsymbol{\omega}}'^{(t)})] = \mathbb{E}[f(\bar{\boldsymbol{\omega}}'^{(t+1)})] - \mathbb{E}[f(\bar{\boldsymbol{\omega}}^{(t+1)})] + \mathbb{E}[f(\bar{\boldsymbol{\omega}}^{(t+1)})] - \mathbb{E}[f(\bar{\boldsymbol{\omega}}'^{(t)})], \tag{25}$$

where the mask update only happens at server, with $\bar{\boldsymbol{\omega}}'^{(t)}$ being the global model received by nodes at round t, at the start of the local model training phase.

Lemma 1. *Given any mask function $\mathbf{m} := \{0,1\}^{n \times p}$ for pruning, the Frobenius norm of model weight/gradients matrix $\boldsymbol{\omega}$ is greater than or equal to the pruned one $\mathbf{m} \odot \boldsymbol{\omega}$, i.e.,*

$$\|\boldsymbol{\omega}\| \geq \|\mathbf{m} \odot \boldsymbol{\omega}\|. \tag{26}$$

Proof. Refer to 'Note for Proofs' at the beginning of this section. □

Lemma 1 ensures that the quality of pruning $\delta_t \in [0,1]$.

Lemma 2. *Given Definition 1 and Assumption 2, the effect of pruning on pruned weights at server (δ_{t+1}) is bounded by:*

$$\mathbb{E}[f(\bar{\boldsymbol{\omega}}'^{(t+1)})] - \mathbb{E}[f(\bar{\boldsymbol{\omega}}^{t+1})] \leq \mu \mathbb{E}[\sqrt{\delta_{t+1}} \|\bar{\boldsymbol{\omega}}^{t+1}\|] \tag{27}$$

Proof. Refer to 'Note for Proofs' at the beginning of this section. □

Lemma 3. *Under Assumptions 1 and 4, $\mathbb{E}[f(\bar{\boldsymbol{\omega}}^{t+1})] - \mathbb{E}[f(\bar{\boldsymbol{\omega}}'^{(t)})]$ is bounded by:*

$$\mathbb{E}[f(\bar{\boldsymbol{\omega}}^{t+1})] - \mathbb{E}[f(\bar{\boldsymbol{\omega}}'^{(t)})] \leq \frac{(\gamma-1)M^2\eta_t^2 + \eta_t^2 M}{2K} \sum_{n=1}^N \rho_n \sigma_n^2 + \frac{(\gamma-1)\gamma L \eta_t^2 M^2}{2} \sum_{k=t+1}^{t+L} \|\sum_{n=1}^N \rho_n \mathbf{v}_n'^{(k)}\|^2 - \frac{\eta_t}{2}\|\nabla f(\bar{\boldsymbol{\omega}}'^{(t)})\|^2 + \frac{\gamma \eta_t^2 M - \eta_t}{2} \|\sum_{n=1}^N \rho_n \tilde{\mathbf{v}}_n'^{(t)}\|^2. \tag{28}$$

Proof. Refer to 'Note for Proofs' at the beginning of this section. □

Theorem 1 (FidDIP Convergence)**.** *Consider the Assumptions 1, 2 and 3, Lemmas 1, 2, 3, and let $\eta_t = \frac{1}{tM}$, $M > 0$. Then, convergence rate of the FedDIP process is bounded by:*

$$\frac{1}{T}\sum_{t=1}^T \|\nabla f(\bar{\boldsymbol{\omega}}'^{(t)})\|^2 \leq 2M\mathbb{E}(f(\boldsymbol{\omega}_1) - f^*) + 2M \sum_{t=1}^T [\mu \mathbb{E}[\sqrt{\delta_{t+1}}\|\bar{\boldsymbol{\omega}}^{t+1}\|] + \frac{\pi^2}{3M^2}\chi, \tag{29}$$

where $f(\boldsymbol{\omega}_1)$ and f^* stand for the initial loss and the final convergent stable loss, with $\chi = \frac{(\gamma-1)M^2+M}{2K}\sum_{n=1}^N \rho_n \sigma_n^2 + \frac{(\gamma-1)\gamma L^2 M^2 G^2}{2}$, and γ defined in Definition 2.

Proof. Refer to 'Note for Proofs' at the beginning of this section. □

In Theorem 1, the first term of the right-hand side of the inequality (29) denotes the gap between the initial and final loss, while χ goes to zero as $K \gg 1$ and the i.i.d. case assumption holds. This also suggests that non-i.i.d. case results in large boundaries. The quantity $\frac{1}{T}\sum_{t=1}^T \|\nabla f(\bar{\boldsymbol{\omega}}'^{(t)})\|^2$ is bounded by the loss produced by pruning. Overall, the convergence result shows that the L_2 norms of the pruned gradients parameters vanish over time, which indicates that a stable model is obtained at the end (recall, a stable gradient vector enables a small change on the model under SGD).

VI. Experimental Evaluation

A. Experimental Setup

Datasets and Models: We experiment with the datasets *Fashion-MNIST* [39], *CIFAR10*, and *CIFAR100* [40]. *Fashion-MNIST* consists of $60,000$ training and $10,000$ test 28x28 grayscale images labeled from 10 classes. Both of *CIFAR* datasets consist of $50,000$ training and $10,000$ test 32x32 colour images; in *CIFAR10* and *CIFAR100* there are 10 classes (6000 images per class) and 100 classes (600 images per class), respectively. We consider the i.i.d. (independent and identically distributed) case to compare all the algorithms and extend FedDIP to be applied for non-i.i.d. cases. To test and compare the efficiency of FedDIP, we use different well-known CNN architectures: *LeNet-5* [41], *AlexNet* [42] and *Resnet-18* [7] as backbone (dense or unpruned) models, with the baseline FedAvg [1] and the pruning baselines PruneFL [12], PruneTrain [26], DPF [27] (equivalent to FedDP as discussed above), and SNIP [23]. For the non-i.i.d. case, we adopt the pathological data partition method in [1], which assigns only two classes for each node. We merge FedDIP with FedProx [43], a generalization and re-parametrization of FedAvg to address the heterogeneity of data (coined FedDIP+Prox), and compare with baseline FedAvg and FedProx. Our target is to evaluate FedDIP's accuracy, storage, and communication efficiency in FL environments under extreme sparsity.

Configurations: Table II details our configurations. For PruneFL and PruneTrain, we experimentally determined the optimal reconfiguration intervals R to be 20 and 1, respectively, to ensure the *best* possible model performance; the same for step size Q for all models. As SNIP prunes the model before training, the global mask is pruned via one-shot achieving the target sparsity s_p. We used grid-search to fix the penalty factor for PruneTrain ranging from 10^{-1} to 10^{-5} for different experiments. When necessary, other hyperparameters were set to match ours. In non-i.i.d. case, the penalty for the proximal term in FedProx is determined via grid-search ranging from 10^{-1} to 10^{-5}. FedDIP+Prox adopts optimal combination of penalty values for FedDIP and FedProx.

Hardware: Our FedDIP framework and experiments are implemented and conducted on *GeForce RTX 3090s* GPUs in the institution's HPC environment.

B. Performance Under Extreme Sparsity

To demonstrate the performance of FedDIP and other baseline methods under extreme sparsity, we set target $s_p = 0.9$ for both *Fashion-MNIST* and *CIFAR10* tasks and $s_p = 0.8$ for the *CIFAR100* task. Notably, as $s_p = 0.9$ causes divergence during the training of *AlexNet* with SNIP, we adjust s_p to 0.8 for SNIP in this particular case.

1) Accuracy: Figures 2a, 3a, and 4a demonstrate that FedDIP surpasses other baselines in achieving the highest *top-1* accuracy (ratio of the correctly classified images) while maintaining the same extreme target sparsity. As indicated in Table III, to attain target sparsity of $s_p = 0.9$ and $s_p = 0.8$ respectively, FedDIP only compromises *LeNet-5* and *ResNet-18* model accuracy by 1.24% and 1.25%, respectively. For *AlexNet*, FedDIP can even improve model performance 0.7%, compared with FedAvg with $s_p = 0.9$.

2) Cumulative Communication & Training Cost: To make a fair comparison of cumulative communication cost during training (amount of information exchanged in MB) w.r.t. a fixed budget, we showcase the relationship between communication cost and accuracy. Figures 2b, 3b, 4b, and specifically Table IV present a comprehensive overview, emphasizing that FedDIP, when provided with adequate communication cost (budget), effectively prunes the model across all experiments outperforming the other models. This indicates the trade-off between model performance and communication/training cost. FedDIP demonstrates comparable communication efficiency to other baselines, principally due to the minimal decrement in model performance. Through our experiments, it is evidenced that FedDIP achieves optimally pruned models under conditions of extreme sparsity, while incurring less or equivalent communication costs compared to FedAvg. Even in the early stages (i.e., in restricted budget cases), FedDIP manages to match the communication efficiency of other pruning methods in the *CIFAR* experiments. This underscores the capacity of our approach to effectively balance model performance and communication expenditure. All in all, FedDIP introduces *only* minor computational overhead due to the incremental regularization, while achieving high accuracy compared to baselines. This computational requirement is on par with that of PruneTrain, PruneFL, and SNIP, given the same sparsity at each epoch. A slight increase in computational cost can be justified by the improvements achieved in the final model performance considering extremely high sparsity. The size of the pruned CNN models (Table III) has been significantly reduced ($\sim$ 1 order of magnitude) from the un-pruned models in FedAvg.

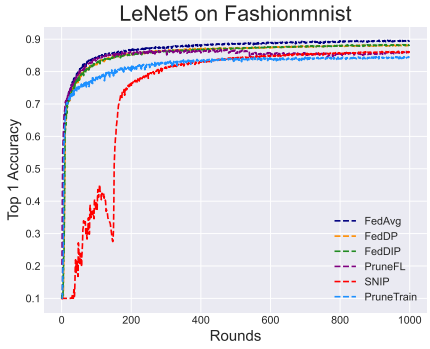
3) Experiments with non-i.i.d. data: As shown in Table V, our methodology exhibits strong adaptability to FedProx (non-pruning), yielding commendable results on non-i.i.d. data. When juxtaposed with FedAvg, our approach manages to maintain comparable results even after pruning 90% of model parameters, albeit at a slight trade-off of 1-2% in model accuracy in the experiments with *LeNet-5* and *AlexNet*. Across a span of $T = 1000$ rounds, FedDIP emerges as the superior performer in terms of *top-1* accuracy, particularly at sparsity $s_p = 0.8$ in *ResNet-18*. This comprehensive suite of results underscores the adaptability of FedDIP in effectively managing non-i.i.d. cases, even in extreme sparsity.

C. FedDIP Sparsity Analysis

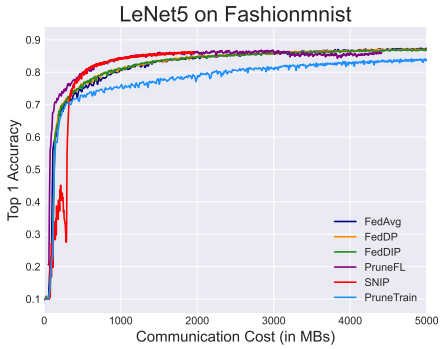
1) Layerwise sparsity: Figure 5 shows the sparsity *per* layer of ResNet-18 ($s_p = 0.8$), LeNet-5 ($s_p = 0.9$), and AlexNet ($s_p = 0.9$). Notably, the first layers of all models are the least pruned ($0.3 \leq s \leq 0.4$), which is attributed to their significant role in general feature extraction. Furthermore, there is a correlation between the number of weights per layer and the corresponding sparsity level. This stems from the initial ERK distribution, which allocates a higher degree of sparsity to layers containing more weights, although we adopt

Datasets	Fashion-MNIST	CIFAR10	CIFAR100
DNN/CNN Model	LeNet-5	AlexNet	ResNet-18
Number of pruning layers (Z)	5	8	18
Initial learning rate (η_0)	0.01	0.1	0.1
Number of clients per round (K)	5 (out of 50)	5 (out of 50)	5 (out of 50)
Batchsize in SGD	64	128	128
Initial sparsity (s_0)	0.5	0.5	0.05
Global rounds (T)	$1,000$	$1,000$	$1,000$
Reconfiguration interval (R)	5	5	5
Regularization step size (Q)	10	10	10
Local round (L)	5	5	5
Maximum penalty ($\lambda_{\max}$)	10^{-3}	10^{-3}	$5 \cdot 10^{-3}$

TABLE II: Configuration Table

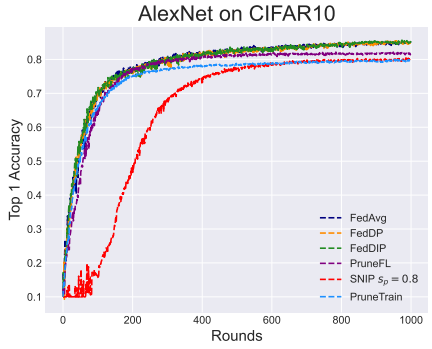


(a) Test accuracy.

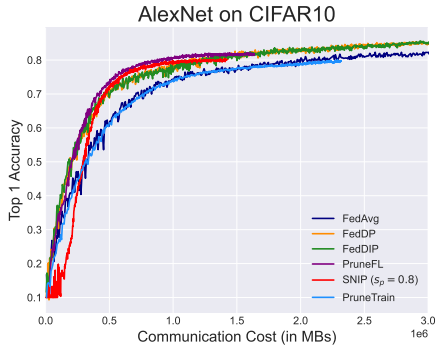


(b) Test accuracy vs. communication budget.

Fig. 2: Fashion-MNIST experiment with LeNet-5.



(a) Test accuracy.

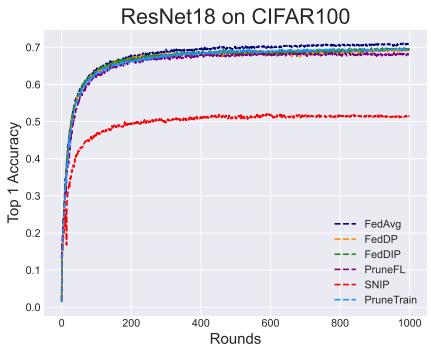


(b) Test accuracy vs. communication budget.

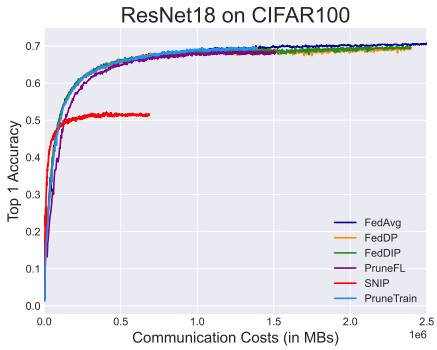
Fig. 3: CIFAR10 experiment with AlexNet.

global magnitude pruning in a later process. Such correlation is remarkable in both convolutional and fully-connected layers of the models. In convolutional layers, the correlations are found to be perfectly linear for *LeNet-5* with a correlation coefficient $\varrho \simeq 1$, for *AlexNet* we obtain $\varrho = 0.86$, while for *ResNet-18* $\varrho = 0.8$. For fully-connected layers, since only one exists in *ResNet-18*, we obtain $\varrho = (0.91, 0.82)$ for *LeNet-5*, *AlexNet*, respectively. These findings highlight the dependency of layerwise sparsity and the number of weights per layer, reflecting the influence of the ERK distribution in FedDIP's initialization.

2) FedDIP in extreme sparsity: We examine the efficiency of FedDIP under varying conditions of extreme sparsity. For *Fashion-MNIST* and *CIFAR10* experiments, we investigate two additional extreme sparsity levels $s_p = 0.95$ and $s_p = 0.99$, and for *CIFAR100* experiments, we investigate $s_p = 0.9$ and $s_p = 0.95$. These conditions provide a robust assessment of FedDIP's performance across a range of extreme sparsity. As shown in Figure 6, under extreme sparsity like 0.95 and 0.99, the largest drops Δ in classification accuracy are only $\Delta = 6.97\%$, $\Delta = 5.03\%$, and $\Delta = 8.08\%$, respectively. This also comes with *further* 90%, 89%, and 74% reduction on LeNet-5, Alex-Net, and ResNet-18 model sizes, respectively. This indicates (i) FedDIP's efficiency in storing and managing trained and pruned models as well as (ii) efficiency in inference tasks (after training) due to relatively small models. All in all, the pruned DNN models' performance is relatively high with small accuracy drops and high model compression (92%)



(a) Test accuracy.



(b) Test accuracy vs. communication budget.

Fig. 4: CIFAR100 experiment with ResNet18.

TABLE III: Test Accuracy (*top-1*)

	Model Performance (%)[1] with target sparsity s_p		
Model	*LeNet*;$s_p = .9$	*AlexNet*;$s_p = .9$	*ResNet*;$s_p = .8$
FedAvg	89.50 (.09)	85.07 (.13)	70.92 (.10)
FedDP	88.06 (.08)	84.81 (.18)	69.23 (.14)
FedDIP	**88.26** (0.09)	**85.14** (.22)	**69.67** (.10)
PruneFL	86.00 (.10)	81.64 (.17)	68.17 (.20)
SNIP	86.08 (.15)	80.10 (.15)	51.46 (.11)
PruneTrain	84.36 (.10)	79.73 (.10)	69.39 (.08)
# param.(FedAvg)	62K	23.3M	11.2M
# param.(pruned)	**6.1K**	**2.3M**	**2.2M**

[1] Mean accuracy; standard deviation in '()'.

across different tasks.

VII. Conclusions

We propose FedDIP, a novel FL framework with dynamic pruning and incremental regularization achieving highly accurate and extremely sparse DNN models. FedDIP gradually regularizes sparse DNN models obtaining extremely compressed models that maintain baseline accuracy and ensure controllable communication overhead. FedDIP is a data-free initialization method based on ERK distribution. We provide a theoretical convergence analysis of FedDIP and evaluate it across different DNN structures. FedDIP achieves comparable and higher accuracy against FL baselines and state-of-the-art FL-based model pruning approaches, respectively, over extreme sparsity using benchmark data sets (i.i.d. & non-i.i.d. cases). Our agenda includes addressing heterogeneity in personalized FL environments.

TABLE IV: Communication Efficiency

	Model Performance (%) with communication budget		
Case	*LeNet-5*$^{(1)}$	*AlexNet*$^{(2)}$	*ResNet-18*$^{(3)}$
FedAvg	86.76	78.98	70.06
FedDP	86.54	82.29	69.10
FedDIP	**86.62**	**82.58**	**69.57**
PruneFL	85.57	81.73	68.4
SNIP	86.32	80.11	51.63
PruneTrain	82.68	78.16	69.42

Communication budget $^{(1)}4 \cdot 10^3$MB, $^{(2)}1.8 \cdot 10^6$MB, $^{(3)}2 \cdot 10^6$MB

TABLE V: Extension to non-i.i.d. data

	Model Performance[1] (%) (non-i.i.d. case)		
Case	*LeNet-5*	*AlexNet*	*ResNet-18*
FedAvg	76.42(0.28)	61.59 (0.73)	16.44 (0.49)
FedProx	**76.63**(0.34)	**65.74** (0.26)	18.48 (0.91)
FedDIP+Prox	74.49 (0.09)	60.47 (0.52)	**19.22** (0.8)

[1] Mean of the highest five *top-1* test accuracy during T rounds.

Acknowledgement

This work is partially funded by the EU Horizon Grant 'Integration and Harmonization of Logistics Operations' TRACE (#101104278).

References

[1] 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.

[2] G. A. Kaissis, M. R. Makowski, D. Rückert, and R. F. Braren, "Secure, privacy-preserving and federated machine learning in medical imaging," *Nature Machine Intelligence*, vol. 2, no. 6, pp. 305–311, 2020.

[3] Q. Yang, Y. Liu, Y. Cheng, Y. Kang, T. Chen, and H. Yu, "Federated learning," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 13, no. 3, pp. 1–207, 2019.

[4] G. Long, Y. Tan, J. Jiang, and C. Zhang, "Federated learning for open banking," in *Federated learning*. Springer, 2020, pp. 240–254.

[5] J. Xu, B. S. Glicksberg, C. Su, P. Walker, J. Bian, and F. Wang, "Federated learning for healthcare informatics," *Journal of Healthcare Informatics Research*, vol. 5, no. 1, pp. 1–19, 2021.

[6] S. Niknam, H. S. Dhillon, and J. H. Reed, "Federated learning for wireless communications: Motivation, opportunities, and challenges," *IEEE Comm. Magazine*, vol. 58, no. 6, pp. 46–51, 2020.

[7] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE CVPR*, 2016, pp. 770–778.

[8] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

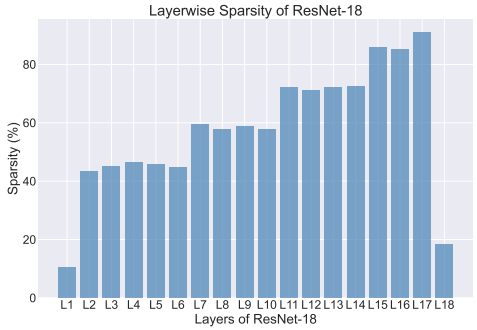
[9] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," 2019.

[10] A. F. Aji and K. Heafield, "Sparse communication for distributed gradient descent," in *EMNLP'17*, 2017, pp. 440–445.

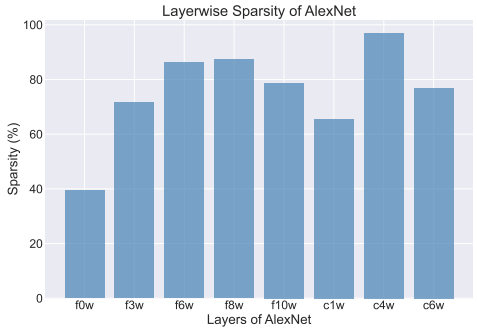
[11] D. Alistarh, T. Hoefler, M. Johansson, N. Konstantinov, S. Khirirat, and C. Renggli, "The convergence of sparsified gradient methods," *NeurIPS'18*, vol. 31, 2018.

[12] Y. Jiang, S. Wang, V. Valls, B. J. Ko, W.-H. Lee, K. K. Leung, and L. Tassiulas, "Model pruning enables efficient federated learning on edge devices," *IEEE TNNLS*, 2022.

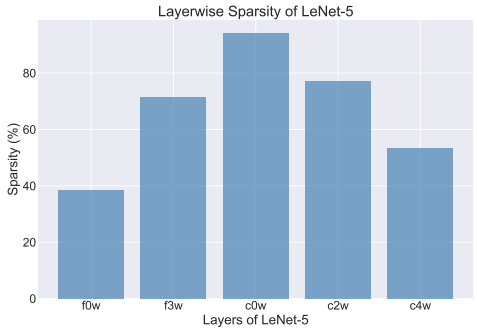
[13] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo, and J. Zhang, "Edge intelligence: Paving the last mile of artificial intelligence with edge computing," *Proc. of IEEE*, vol. 107, no. 8, pp. 1738–1762, 2019.



(a) Distribution of layer sparsity; ResNet-18.



(b) Distribution of layer sparsity; AlexNet.



(c) Distribution of layer sparsiy; LeNet-5.

Fig. 5: Layerwise pruning sparsity; *f0w* stands for (*f*)eatures layer, layer index (e.g, 0), and (*w*)eights, respectively. *c* stands for the fully-connected classifier layer (the same notation is used for other layers). ResNet-18 consists of 18 pruning layers.

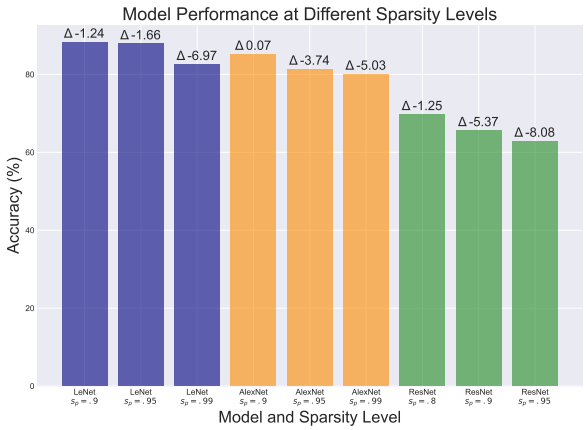


Fig. 6: FedDIP performance on extreme sparsity values.

[14] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Proc. Magazine*, vol. 37, no. 3, pp. 50–60, 2020.

[15] N. Strom, "Scalable distributed dnn training using commodity gpu cloud computing," in *16th Intl Conf Speech Comm. Assoc.*, 2015.

[16] J. Konečnỳ and P. Richtárik, "Randomized distributed mean estimation: Accuracy vs. communication," *Frontiers in Applied Mathematics and Statistics*, vol. 4, p. 62, 2018.

[17] P. Jiang and G. Agrawal, "A linear speedup analysis of distributed deep learning with sparse and quantized communication," *NeurIPS'18*, vol. 31, 2018.

[18] Y. Lin, S. Han, H. Mao, Y. Wang, and W. J. Dally, "Deep Gradient Compression: Reducing the communication bandwidth for distributed training," in *ICLR*, 2018.

[19] S. Shi, K. Zhao, Q. Wang, Z. Tang, and X. Chu, "A convergence analysis of distributed sgd with communication-efficient gradient sparsification." in *IJCAI*, 2019, pp. 3411–3417.

[20] J. Sun, T. Chen, G. Giannakis, and Z. Yang, "Communication-efficient distributed learning via lazily aggregated quantized gradients," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[21] T. Chen, G. Giannakis, T. Sun, and W. Yin, "Lag: Lazily aggregated gradient for communication-efficient distributed learning," *NeurIPS'18*, vol. 31, 2018.

[22] Y. Cheng, D. Wang, P. Zhou, and T. Zhang, "A survey of model compression and acceleration for deep neural networks," *arXiv preprint arXiv:1710.09282*, 2017.

[23] N. Lee, T. Ajanthan, and P. H. Torr, "Snip: Single-shot network pruning based on connection sensitivity," *ICLR 2019 arXiv preprint arXiv:1810.02340*, 2018.

[24] Y. He, X. Zhang, and J. Sun, "Channel pruning for accelerating very deep neural networks," in *IEEE ICCV*, 2017, pp. 1389–1397.

[25] T. Zhang, S. Ye, K. Zhang, J. Tang, W. Wen, M. Fardad, and Y. Wang, "A systematic dnn weight pruning framework using alternating direction method of multipliers," in *ECCV*, 2018, pp. 184–199.

[26] S. Lym, E. Choukse, S. Zangeneh, W. Wen, S. Sanghavi, and M. Erez, "Prunetrain: fast neural network training by dynamic sparse model reconfiguration," in *SC'19*, 2019, pp. 1–13.

[27] T. Lin, S. U. Stich, L. F. Barba Flores, D. Dmitriev, and M. Jaggi, "Dynamic model pruning with feedback," in *ICLR*, no. CONF, 2020.

[28] X. Yao, T. Huang, R.-X. Zhang, R. Li, and L. Sun, "Federated learning with unbiased gradient aggregation and controllable meta updating," *arXiv preprint arXiv:1910.08234*, 2019.

[29] A. Li, J. Sun, B. Wang, L. Duan, S. Li, Y. Chen, and H. Li, "Lotteryfl: empower edge intelligence with personalized and communication-efficient federated learning," in *IEEE/ACM SEC*. IEEE, 2021, pp. 68–79.

[30] U. Evci, T. Gale, J. Menick, P. S. Castro, and E. Elsen, "Rigging the lottery: Making all tickets winners," in *ICML*. PMLR, 2020, pp. 2943–2952.

[31] M. Zhu and S. Gupta, "To prune, or not to prune: exploring the efficacy of pruning for model compression," *arXiv preprint arXiv:1710.01878*, 2017.

[32] H. Wang, C. Qin, Y. Zhang, and Y. Fu, "Neural pruning via growing regularization," in *ICLR*, 2021.

[33] W. Fu and K. Knight, "Asymptotics for lasso-type estimators," *The Annals of Statistics*, vol. 28, no. 5, pp. 1356–1378, 2000.

[34] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," *PMLR*, vol. 2, pp. 429–450, 2020.

[35] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *NeurIPS'19*. Curran Associates, Inc., 2019, pp. 8024–8035.

[36] F. Haddadpour and M. Mahdavi, "On the convergence of local descent methods in federated learning," *CoRR*, vol. abs/1910.14425, 2019. [Online]. Available: http://arxiv.org/abs/1910.14425

[37] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of fedavg on non-iid data," in *ICLR*, 2019.
[38] S. Wan, J. Lu, P. Fan, Y. Shao, C. Peng, and K. B. Letaief, "Convergence analysis and system design for federated learning over wireless networks," *IEEE JSAC*, vol. 39, no. 12, pp. 3622–3639, 2021.
[39] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017.
[40] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.
[41] Y. LeCun *et al.*, "Lenet-5, convolutional neural networks," *URL: http://yann. lecun. com/exdb/lenet*, vol. 20, no. 5, p. 14, 2015.
[42] A. Krizhevsky, "One weird trick for parallelizing convolutional neural networks," *CoRR*, vol. abs/1404.5997, 2014. [Online]. Available: http://arxiv.org/abs/1404.5997
[43] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," in *PMLS*, vol. 2, 2020, pp. 429–450.