Automated Federated Learning via Informed Pruning

Christian Internò[1,3] **Elena Raponi**[2] **Niki van Stein**[2] **Thomas Bäck**[2] **Markus Olhofer** [3]
Yaochu Jin [4] **Barbara Hammer** [1]

[1]CITEC, University of Bielefeld, Bielefeld, Germany
[2]LIACS, Leiden University, Leiden, Netherlands
[3]Honda Research Institute EU, Offenbach, Germany
[4]Westlake University, Hangzhou, Zhejiang, China

Abstract Federated learning (FL) represents a pivotal shift in machine learning (ML) as it enables collaborative training of local ML models coordinated by a central aggregator, all without the need to exchange local data. However, its application on edge devices is hindered by limited computational capabilities and data communication challenges, compounded by the inherent complexity of Deep Learning (DL) models. Model pruning is identified as a key technique for compressing DL models on devices with limited resources. Nonetheless, conventional pruning techniques typically rely on manually crafted heuristics and demand human expertise to achieve a balance between model size, speed, and accuracy, often resulting in sub-optimal solutions.

In this study, we introduce an automated federated learning approach utilizing informed pruning, called `AutoFLIP`, which dynamically prunes and compresses DL models within both the local clients and the global server. It leverages a federated loss exploration phase to investigate model gradient behavior across diverse datasets and losses, providing insights into parameter significance. Our experiments showcase notable enhancements in scenarios with strong non-IID data, underscoring `AutoFLIP`'s capacity to tackle computational constraints and achieve superior global convergence.

1 Introduction

In the past decade, the proliferation of smart devices at the network edge and the surge in data generated and distributed across them have created a decentralized setting. Here, multiple participants store their data locally, which offers an opportunity for collaborative model training, enhancing robustness and generalization. Distributing the computational load across these devices results in faster training times and lower energy consumption compared to centralized approaches.

Nonetheless, distributed machine learning (ML) training encounters significant challenges. One major challenge is ensuring efficient communication and coordination among multiple participants, as each device typically holds only a subset of the data. Effective strategies for aggregating and updating model parameters without excessive communication overhead are crucial. This involves designing algorithms that minimize data exchange while enabling the convergence of high-quality models. Moreover, device heterogeneity, encompassing differences in computational power, storage capacity, and network bandwidth, complicates distributed ML training. Developing algorithms capable of adapting to such environments and utilizing resources efficiently is essential for scaling up distributed learning to large-scale deployments.

Privacy and security concerns are another challenge (Hoffpauir et al., 2023). With sensitive data distributed across various devices, ensuring the privacy of individual data points becomes paramount. For example, medical data stored in hospitals and personal devices is valuable for training diagnostic models but is also subject to strict privacy and security regulations. In this context, Federated Learning (Zhu et al., 2021) emerges as an effective strategy for training always

 © 2024 the authors, released under CC BY 4.0

more complex deep learning (DL) models while preserving the privacy of the data. FL facilitates collaborative model training across multiple devices without exposing local data. A central server, i.e., a global model, coordinates this process by aggregating the updates from locally trained models, which ensures a secure learning environment.

Current FL research primarily focuses on enhancing privacy and adapting ML workflows for specific uses, often with predetermined ML model configurations. For instance, tasks related to computer vision may involve centralized servers selecting well-known neural network architectures like VGG-16 (Simonyan and Zisserman, 2015) (with approximately 138 million parameters) or ResNet-50 (He et al., 2016) (with around 25.6 million parameters). However, the high complexity of these neural networks increases the risk of overfitting, especially with small training data sizes. Additionally, FL systems typically expect clients to have access to high-speed processors and sufficient computational power for local calculations and parameter updates. Yet, many edge devices, such as smartphones, wearables, and sensors, have limited computing and memory capacities, posing a challenge to DL model training systems (Hoffpauir et al., 2023). While DL networks have significantly improved in performance and accuracy, their larger model sizes and increased computational costs present challenges for edge devices that struggle to handle such demands. Moreover, the communication of DL models with millions of parameters presents significant obstacles for FL transmission (Shlezinger et al., 2020; Asad et al., 2023). Therefore, the effective use of FL with edge devices that have limited computational capabilities, while maintaining efficient communication, is an active research question. This underscores the need for personalized and innovative approaches in FL, particularly in optimizing and compressing models to improve inference time, communication cost, energy efficiency, and complexity, all while maintaining a satisfactory accuracy (Rahman et al., 2021).

Numerous methods have been suggested to enhance the effectiveness of neural networks through model compression (Li et al., 2023). Traditionally, this process involves the use of manual rules and specialized expert knowledge to navigate the vast design space, balancing the size, speed and accuracy of the model. However, the vast design space often renders human heuristics a time-consuming strategy for model compression, often leading to suboptimal solutions. Automated Machine Learning (AutoML) systems, like autosklearn (Feurer et al., 2022), TPOT (Olson and Moore, 2016), and H2O AutoML (H2O.ai, 2022), streamline model selection and optimization in centralized settings. Yet, in FL, these systems face challenges due to the distributed nature of data and the impracticality of centralizing it for model selection, given privacy or resource constraints. This backdrop underscores the necessity for novel methods that optimize DL models in FL without centralizing client data, taking into account the available limited resources, the privacy of sensitive information, and addressing class imbalance among the clients in non-IID scenarios.

Our contribution. We introduce a novel automated federated learning approach via informed pruning (`AutoFLIP`), which uses a loss exploration mechanism (Nikolić et al., 2023) to automatically prune and compress DL models based on shared insights into gradient behaviors across clients. This strategy allows for dynamically reducing the complexity of the DL models in FL environments, hence optimizing performance with limited computational resources at the client level. In our assumed single-server architecture, each client operates on the same initial deep neural network structure that automatically prunes itself at each round, based on the extraction of shared knowledge for an informed model compression. With our experiments over various datasets, tasks, and realistic non-IID scenarios, we provide strong evidence of the effectiveness of the proposed method.

Reproducibility. Our code for reproducing the experiments is available on GitHub.[1]

[1] `https://github.com/ChristianInterno/AutoFLIP`

2 Background and Related Work

2.1 Federated Learning

One widely accepted FL standard is known as *FedAvg* (McMahan et al., 2023). It operates on a distinctive collaborative principle. The global model disseminates a model to different clients for localized training. The impact of each client on the global model update is determined by the size of its dataset, ensuring that clients with larger datasets have a more significant influence on the final global model. This is accomplished by calculating the server parameters as a weighted average of the individual parameters learned by each client. This iterative process, characterized by collaborative learning dynamics, continues until the model reaches an satisfactory termination criterion. Empirical studies have shown the robustness of this approach, even when handling non-convex optimization problems (Das et al., 2022). As a result, it is commonly used as a standard for evaluating newly developed FL protocols. In this study, we will compare the performance of the proposed `AutoFLIP` method to FedAvg, with and without random pruning, as tested in (Wu et al., 2023). Indeed, among various parameter selection criteria, random pruning demonstrated rapid convergence within their proposed framework, outperforming any other pruning heuristic.

2.2 Model Pruning

Following the assumption that a DL model can contain a sub-network that represents the performance of the entire model after being trained, model pruning is a good strategy to reduce computational requirements of resource-constrained devices. First attempts of using pruning to deploy deep neural networks on resource-limited devices have considered pre-trained convolutional neural networks in a centralized setting (You et al., 2019; Lin et al., 2020). After this, there has also been work focused on dynamic active pruning to increase communication efficiency and reduce the computation cost on the clients during training (Liu et al., 2022; Zhou et al., 2021). Nonetheless, this increases the amount of calculations at the client level. Jiang et al. (2023) introduce `PruneFL`, a FL method that incorporates adaptive and distributed parameter pruning. Their approach utilizes an unstructured method that does not take advantage of the collective insights of participating clients to develop a cooperative structured pruning strategy. This is in contrast to the objectives of `AutoFLIP`, which seeks to harness client-specific knowledge to facilitate a structured approach to pruning. Lin et al. (2022) introduced a novel approach for adaptive per-layer sparsity, however without incorporating any parameter aggregation scheme to reduce the error caused by pruning. This challenge was addressed by Tingting et al. (2023), who were the first applying model pruning directly into a federated learning framework. They moved the pruning process to the global model that works on a computationally more powerful server. The pruned model is distributed to each client, where it undergoes training. Subsequently, each client sends back to the server only the updated parameters, restoring the full structure of the model at the server. Although this study includes various parameter selection criteria from the literature, its pruning method does not incorporate the information gathered during model training. This contrasts with our strategy, `AutoFLIP`, which leverages such information to enhance the pruning process. The work by Yu et al. (2021) explores enhancing FL with an adaptive pruning method to remove less important parameters, but this adds complexity and overhead. Yu et al. (2023) then proposed Resource-aware Federated Foundation Models, focusing on integrating large transformer-based models into FL, with the limitation of not exploring other architectures. Our method, `AutoFLIP`, diverges by introducing a pruning strategy that avoids the need for continuous evaluation of parameter significance and is universally applicable across various FL aggregation algorithms and model architectures.

3 Methodology

`AutoFLIP` is an automatized FL approach relying on informed pruning. Preliminary step to the FL rounds is an initial exploration phase. Here, a portion or the totality of the clients, which inherit

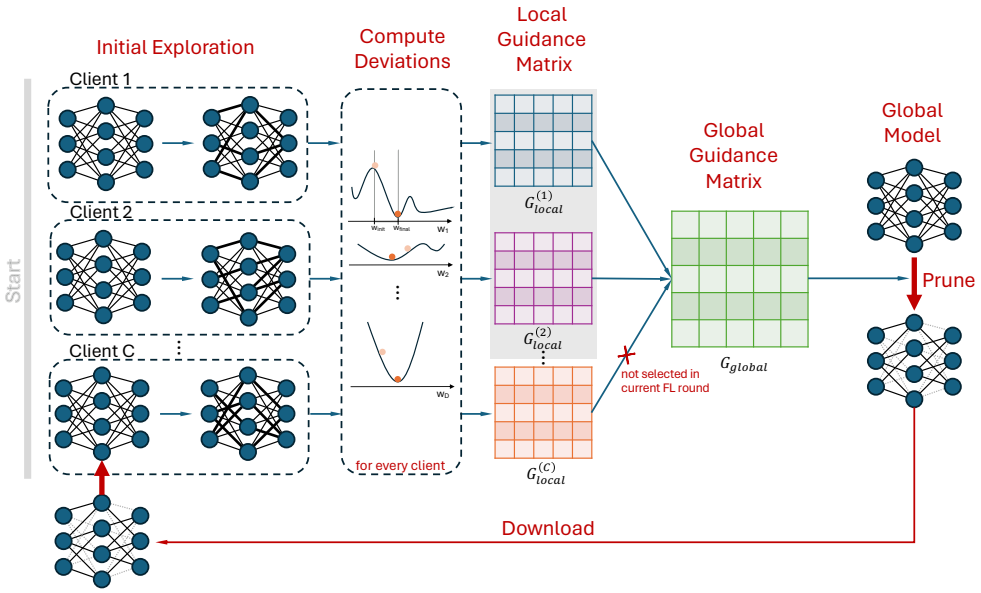

Figure 1: Illustration of the pruning procedure. The local guidance matrices are computed a priori through an exploration phase. The global guidance matrix is computed aggregating the elements of the local guidance matrices corresponding to the clients participating in each FL round. The global model is pruned and the reduced model is downloaded by the clients.

their topology from the global model, are trained for a certain number of exploration epochs. Based on this, for each client, we compute a local guidance matrix, which records how important a certain parameter (weight or bias) is in terms of parameter deviation during the exploration, which reflects loss variability. Afterward, we aggregate the information collected locally in a global guidance matrix, which guides the pruning of the global model.

In each FL round, only a subset of clients influence the global model pruning. The global guidance matrix is recalculated by summing entries from local matrices of selected clients. Using the updated global guidance matrix, the global model pruned parameters are adjusted. Subsequently, the reduced model is distributed to participating clients, who train their pruned models. Server parameters are computed as a weighted average of individual client parameters. The iterative procedure, consisting of (1) pruning of the global and local models, (2) training of the reduced local models, (3) computation of the global model parameters, and (4) local and global model performance evaluation, continues until a prescribed termination criterion is met.

The pruning workflow of `AutoFLIP` is illustrated in Figure 1. Please note that the initial exploration, computation of weight deviations, and definition of local guidance matrices occur only once. In contrast, the global guidance matrix and subsequent pruning are redefined at each FL round, considering the clients participating in that round.

In the following, we present the notation used to formalize the above procedure and delve deeper into the problem formulation, exploring specific phases of the `AutoFLIP` workflow.

Notation. We consider a total number of C clients. At each FL round, K clients are chosen and trained on different batches of size B at each epoch, for a total amount of E epochs. The total number of rounds is R, which represents our termination criterion. For the exploration phase, we denote with C_{exp} the number of clients selected, which, in this study, we take as the totality C of available clients. The exploration lasts for E_{exp} epochs. For our informed pruning technique, we define a threshold T_p for the parameter deviations in the exploration.

Problem Definition and Objective. In FL, the main task is to train a global model across C clients, each having a distinct dataset $D_i = \{(\mathbf{x}_i, \mathbf{y}_i)\}$ of size n_i in a non-IID environment with limited computational resources. This means that each D_i is sampled independently from a distribution P_i on both input and label spaces. Each client model has its own parameters W_i, but on the same DL model architecture. The expected loss for each client model can be defined as $L_{P_i}(h_i, W_i) = \mathbb{E}_{(\mathbf{x}_i,\mathbf{y}_i)\sim P_i}[l(h_i(\mathbf{x}_i; W_i), \mathbf{y}_i)]$, where L_{P_i} denotes the expected loss over the data distribution P_i for client c_i with hypothesis h from the model space H, encompassing all possible models.

Our objective is to automatically determine the best structured pruning strategy to significantly compress client models while aligning the global model parameters W_{global} with the optimal W^*_{global} that minimizes the combined loss:

$$\min_{h_i \in H} \frac{1}{C} \sum_{i=1}^{C} L_{P_i}(h_i, W_{i,t}), \tag{1}$$

where $W_{i,t}$ represents the parameters of client c_i at iteration t.

Pruning with AutoFLIP. In `AutoFLIP`, the model initialization phase is augmented by a crucial loss federated exploration phase, allowing clients to explore their loss function landscapes. This exploration phase is completed for embedding gradient variability information into a Pruning Guidance mask PG_{global}. This mask is updated and shared among participating clients in each FL round to guide the evolution of client model structures within an informed pruning session.

To construct the mask PG_{global}, we need an initial exploration phase conducted on C_{exp} clients. In this study, we consider $C_{exp} = C$. We train the clients for E_{exp} epochs, and, for each model parameter, we evaluate its evolution in the search space during the loss exploration phase. This evaluation is conducted by calculating the deviation $D_{1,m}$ for the m^{th} parameter of a client model c_i as the squared difference between the initial ($W^{\text{Initial}}_{i,m}$) and final ($W^{\text{Final}}_{i,m}$) parameter values, following E_{exp} epochs of exploration:

$$D_{i,m} = (W^{\text{Initial}}_{i,m} - W^{\text{Final}}_{i,m})^2. \tag{2}$$

Given that we use stochastic gradient descent for the exploration, this deviation acts as a measure of loss variability dependent on the parameter m. The greater the variation in the parameter space, the more significant the improvement in loss. The C_{exp} clients compile these deviations into a local matrix G_{local}, whose entries are the deviations for the model parameters.

At each FL round, where only K clients are involved, the server aggregates the G_{local} matrices associated to those client to formulate G_{global} through a normalization process:

$$G_{\text{global}} = \frac{1}{K} \sum_{k=1}^{K} \frac{G_{local_k} - \min(G_{\text{local}})}{\max(G_{\text{local}}) - \min(G_{\text{local}})} \tag{3}$$

Each element of G_{global} thus represents the mean normalized deviation for each parameter, scaled between 0 and 1. A value closer to 0 indicates minimal deviation, suggesting gradient stability during the exploration, hence scarce relevance of the parameter itself. Conversely, values near 1 highlight significant parameter deviations, pointing to more dynamic and potentially insightful areas of the loss landscape. Then, a binarization process is then applied to G_{global} where elements below T_p are set to 0 and those above are set to 1:

$$PG_{\text{global},m} = \begin{cases} 0 & \text{if } G_{\text{global},m} < T_p \\ 1 & \text{otherwise} \end{cases} \tag{4}$$

Given their smaller influence, parameters corresponding to 0 are flagged for pruning, whereas those marked with 1 are retained, indicating important search directions within the model parameter

space. During each FL round, the K participating clients update PG_{global} by incorporating their G_{local} deviation values derived from the initial loss exploration phase.

Algorithm 1 `AutoFLIP` Algorithm

1: **Server Initialization**: Initial weight matrix $W_{global}^{(0)}$, number of clients for exploration C_{exp}, exploration epochs E_{exp}, pruning threshold T_p, FL rounds R, training epochs E, number of selected clients per round K
2: Server selects C_{exp} clients for exploration
3: $G_{local_i} = (W_i^{\text{Initial}} - W_i^{\text{Final}})^2, \forall i \in [1, C_{exp}]$
4: **for** round $r = 1$ to R **do**
5: Server selects K clients
6: Compute $G_{\text{global}}^{(r)}$ using Eq. (3)
7: Compute mask $PG_{\text{global}}^{(r)}$ using Eq. (4)
8: **for** client $k = 1$ to K **do**
9: $W_{k,\text{pruned}}^{(r)} = W_k^{(r)} \odot PG_{\text{global}}^{(r)}$
10: **for** each local epoch $e = 0$ to $E - 1$ **do**
11: $W_{k,\text{pruned}}^{(e+1)} = W_{k,\text{pruned}}^{(e)} - \eta \nabla L_k \left(W_{k,\text{pruned}}^{(e)} \right)$
12: **end for**
13: **end for**
14: $W_{global}^{(r)} = \frac{1}{K} \sum_{k=1}^{K} W_{k,\text{pruned}}^{(E)}$
15: **end for**

Federated Learning with AutoFLIP. Here our aim is to argument how the parameter pruning mechanism based on loss exploration enters a general FL edge training framework. Algorithm 1 provides an overview of the entire framework of the proposed `AutoFLIP` algorithm for FL. It is composed by the following steps.

Server initialization (Line 1). The server is initialized with a global model that it is sent to all the clients. At this stage, the total number of clients undergoing exploration, the number of exploration epochs, and the pruning threshold are also decided.

Exploration phase (Lines 2–3). The preliminary exploration phase aimed at understanding the relevance of each parameter (weight or bias) in view of loss improvement starts. For each client participating (in this study we select all the available clients), a local guidance matrix storing parameter deviations is computed.

Mask update (Lines 5–7). A FL round starts. The server selects K clients that participate in the round. Only the local guidance matrices of those clients are considered to compute a global guidance matrix, which is then used to generate a binary mask for pruning. The mask contains ones only for the parameters with normalized deviations higher than a prescribed threshold T_p.

Pruning (Lines 8–9). During each round, clients use the pruning mask to compress their models. This happens through element-wise multiplication between their weight matrix and PG_{global} at that FL round. Parameters aligned with a 0 in PG_{global} are pruned; those corresponding to a 1 are kept.

FL round with reduced client models (Lines 10–14). The standard algorithm FedAvg (McMahan et al., 2023) is used on the reduced framework. The pruned clients are trained. The server receives the local model updates and, upon aggregation, proceeds to update the global model with the FL aggregation strategy. Once updated, the global model is either ready for the next communication round or deemed ready for deployment if the convergence criteria are satisfied.

Accuracy-Improved Model Compression. A key aspect of `AutoFLIP` is its ability to maintain or potentially improve model accuracy during compression. Its iterative updating technique,

powered by automated parameter importance evaluation, continuously refines the compression strategy, ensuring alignment with evolving data distributions and learning objectives. In non-IID data environments, AutoFLIP adapts its compression approach to suit each client's dataset characteristics, preventing disproportionate impact on clients with less represented data features and safeguarding overall model accuracy across the network.

The global pruning mask, PG_{global}, is designed to systematically minimize the variance of the trained parameters across different clients, $\sigma^2_{\Delta W} = \text{Var}[\Delta W_1, \Delta W_2, \ldots, \Delta W_K]$. As shown in Figure 2, we prune the parameters that are characterized by scarce variability during the exploration phase and are therefore heavily influenced by their initialization value. Conversely, we continue to include in training the parameters that exhibit the highest deviations during the exploration phase. This indicates that their evolution was marked by high gradients, resulting in rapid exploration over similar loss landscapes (given that we are addressing similar prediction tasks).

By reducing variance through the preliminary exploration procedure, PG_{global} crucially improves the convergence efficiency of the global model. This ensures that the global model evolves efficiently while aligning with the collective learning objectives of all participating clients.

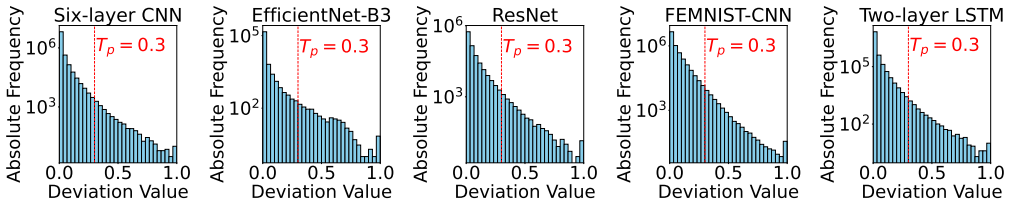

Figure 2: Distribution of parameter deviations in G_{global} after exploration. Absolute frequency in log-scale is shown for each normalized deviation. Higher frequencies are recorded for smaller deviation values, indicating that many parameters are irrelevant for loss improvement.

Communication efficiency. AutoFLIP enhances communication efficiency in FL by reducing model sizes transmitted between clients and the server, thus lowering bandwidth requirements for FL rounds. Its selective updating mechanism ensures only essential parameters, those significantly affecting model performance, are communicated. Integrating AutoFLIP into FL systems enhances faster training and inference times, lower energy consumption, and improved model scalability. The appendices D, E and F provide a detailed analysis of how AutoFLIP accelerates inference and improves training efficiency, demonstrating its significant role in lowering computational costs and boosting FL's overall efficiency.

4 Experiments

Inspired by Hahn et al. (2022), we benchmark AutoFLIP across established datasets to evaluate its robustness and adaptability in various Non-IID environments. We explore three distinct partitioning approaches for creating strongly Non-IID conditions: a **Pathological Non-IID scenario**, which involves 20 clients, each using data from two distinct classes, employing MNIST with a six-layer CNN (7,628,484 parameters) and CIFAR10 with EfficientNet-B3 (10,838,784 parameters), a **Dirichlet-based Non-IID scenario**, which utilizes the Dirichlet distribution to distribute data among 20 clients, with varying class counts per client, using CIFAR100 and a ResNet architecture (23,755,900 parameters), and a **LEAF Non-IID scenario**, which adopts the LEAF benchmark with FEMNIST and Shakespeare datasets. For FEMNIST, a CNN architecture with 13,180,734 parameters is used. For Shakespeare, we consider a two-layer LSTM model with 5,040,000 parameters. Further details on these scenarios are provided in Appendix B.

4.1 Experimental Setup

We evaluate AutoFLIP against FedAvg and different SOTA federated pruning strategies, including Random, L1, L2, Similarity, and BN mask, as described in Wu et al. (2023). We use 20 clients, a batch size of 350, and a learning rate of 0.0003 over 200 FL rounds involving 5 clients each. The setup includes a server momentum of 0.9, an SGD optimizer with weight decay, up to 150 exploration epochs, and a pruning threshold of 0.3. Data is split into 80-20 for training and testing, with model performance assessed by average prediction accuracy on test sets. We run 10 experiment repetitions to ensure statistical validity, measuring the compression rate to gauge model size reduction and its impact. We conducted experiments on two workstations, one with an Intel Core i5 and 16 GB RAM, and another with an Intel Xeon X5680, 128 GB of DDR4 RAM, and an NVIDIA TITAN X GPU.

4.2 Results

In Figure 3, we show the accuracy convergence profiles for the global model. The different federated pruning strategies are evaluated on both image recognition and text prediction tasks using five distinct datasets: MNIST, CIFAR10, CIFAR100, FEMNIST, and Shakespeare. Due to the varying complexities of each task, we use different model structures for different datasets. A plot depicting the evolution of the loss metric is presented in Appendix C.

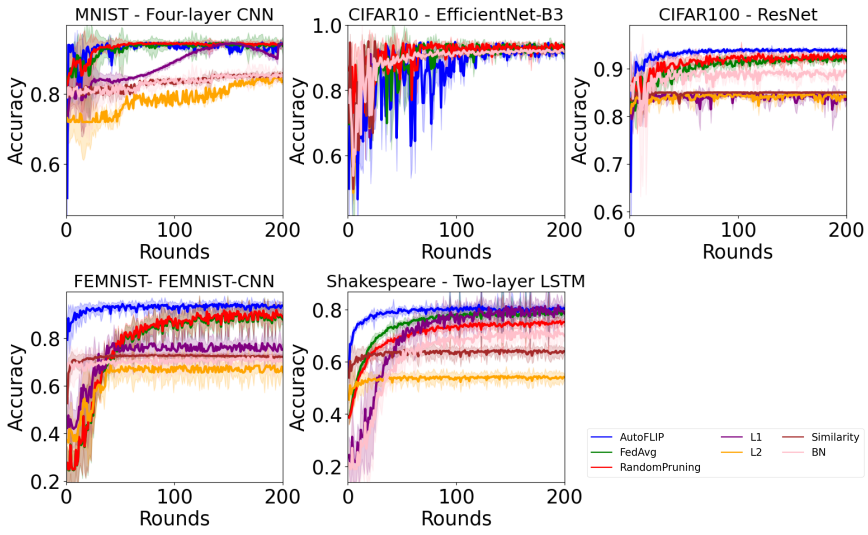

Figure 3: Average accuracy convergence profiles for the global model within the FL framework.

Pathological Non-IID. Here, AutoFLIP achieves an average client compression rate of x1.74. At each round, we remove on average 3244298 parameters for the six-layer CNN. For the EfficientNet-B3, we obtain an average compression rate of x2.1 with 5677458 deleted parameters. For the baselines we compare AutoFLIP to, we ensure that the number of parameters pruned matches the compression ratio of AutoFLIP, quantified as 42% for the six-layer CNN and 52.38% for EfficientNet-B3.

The first two subplots in Figure 3 show the evolution of global model accuracy during the FL rounds for the four-layer CNN with the MNIST dataset and for EfficientNet-B3 with the CIFAR10 dataset. In the case of the MNIST, the early rounds of FL show that AutoFLIP achieves slightly higher accuracy compared to both FedAvg and the other federated pruning strategies, among which RandomPruning emerges as the top performer. This indicates a faster convergence rate for our proposed method. However, the performance of the three baselines soon becomes comparable, with no clear superiority as the FL procedure progresses. We attribute this to the simplicity of the prediction tasks on the MNIST dataset compared to the excessive complexity of the four-layer CNN,

which already possesses extremely good prediction capabilities that cannot be further enhanced by pruning. For the CIFAR10 dataset, we do not observe any advantage in using AutoFLIP over the other baselines. Surprisingly, all methods exhibit severe fluctuations in the accuracy convergence profiles up to FL round 100, after which they stabilize and become comparable.

Dirichlet-based Non-IID. For ResNet, AutoFLIP achieves an average compression rate for the clients of x1.58, with 8,720,520 parameters pruned on average out of 23,755,900 total parameters. Hence, we adjust the percentage of parameters to be pruned to 36.71% for the different baselines.

The third subplot in Figure 3 illustrates the evolution of the global model accuracy during the FL rounds for ResNet on CIFAR100. Here, `AutoFLIP` exhibits a performance enhancement throughout the considered training rounds. At round 200, it achieves an accuracy of 0.987, compared to 0.918 for FedAvg and 0.925 for RandomPruning. This enhancement signifies the robustness of `AutoFLIP`, showcasing its ability to maintain elevated performance levels, even when integrated with larger-complex neural networks and larger datasets.

LEAF Non-IID. In this scenario, `AutoFLIP` achieves an average compression rate of x1.8 for 5858104 client parameters pruned out of 13180734. Hence, we adjust the number of parameters to be pruned for the different baselines to 44%. As observed in the last two subplots of Figure 3 for the FEMNIST and Shakespeare datasets, `AutoFLIP` consistently outperforms the other pruning strategies by a significant margin. What stands out is the initial acceleration in convergence speed observed for `AutoFLIP`, firmly establishing it as a superior choice over FedAvg, RandomPruning and the other baselines in terms of maintaining limited computation and communication costs (see Appendices D and E). Furthermore, this superiority persists throughout the entire FL training procedure. The final average accuracy values are 0.985 for `AutoFLIP`, 0.905 for FedAvg, and 0.935 for RandomPruning on the FEMNIST dataset. For the Shakespeare dataset, the values are 0.815, 0.783, and 0.738, respectively. Here, even L1 proves to be competitive, reaching a final accuracy equal to 0.802. However, it demonstrates inferior initial convergence. The remarkable performance of `AutoFLIP` in these tasks underscores its effectiveness in handling complex and realistic FL scenarios, thus emphasizing its applicability and robustness in real-world settings.

5 Conclusion and future perspectives

We presented AutoFLIP, an automated federated learning (FL) approach via informed pruning. Our method adjusts client DL model compression through informed pruning, leveraging information from a preliminary federated loss exploration phase. This enables efficient global model training and inference on clients with limited computational resources, such as network edge devices.

Extensive experiments across various non-IID scenarios show AutoFLIP's ability to achieve high model accuracy while reducing computational and communication overhead. It improves convergence rates in federated settings with non-IID data distributions compared to FedAvg, with or without random pruning. AutoFLIP demonstrates adaptability and scalability across DL network types and multi-class datasets, showing notable improvements as task complexity increases. This underscores its potential for future research avenues, such as leveraging loss exploration for guiding more complex neural architecture search tasks.

While AutoFLIP has achieved promising results, we recognize that our analysis is not yet complete. Firstly, although to our knowledge we are the first to introduce *automated informed* pruning in an FL framework, we plan to conduct extensive comparisons with other SOTA pruning methods for FL. Additionally, validating AutoFLIP across various real-world domains such as healthcare, Internet of Things, and mobile computing could further demonstrate its effectiveness across different federated environments and data modalities.

6 Broader Impact Statement

After careful reflection, the authors have determined that this work presents no notable negative impacts on society or the environment. On the contrary, by introducing an automatic informed pruning strategy for federated learning, the method presented in this study offers a more sustainable and efficient utilization of edge devices in federated learning tasks.

References

Acknowledgements.

Asad, M., Shaukat, S., Hu, D., Wang, Z., Javanmardi, E., Nakazato, J. and Tsukada, M. (2023), 'Limitations and future aspects of communication costs in federated learning: A survey', *Sensors* **23**(17).

Caldas, S., Duddu, S. M. K., Wu, P., Li, T., Konečný, J., McMahan, H. B., Smith, V. and Talwalkar, A. (2019), 'Leaf: A benchmark for federated settings'.

Das, R., Acharya, A., Hashemi, A., Sanghavi, S., Dhillon, I. S. and Topcu, U. (2022), Faster non-convex federated learning via global and local momentum, *in* 'Conference on Uncertainity in Artificial Intelligence (UAI) (UAI)'.

Deng, L. (2012), 'The mnist database of handwritten digit images for machine learning research [best of the web]', *IEEE Signal Processing Magazine* .

Feurer, M., Eggensperger, K., Falkner, S., Lindauer, M. and Hutter, F. (2022), 'Auto-sklearn 2.0: hands-free automl via meta-learning', *J. Mach. Learn. Res.* **23**(1).

H2O.ai (2022), *h2o: Python Interface for H2O*. Python package version 3.42.0.2.

Hahn, S.-J., Jeong, M. and Lee, J. (2022), Connecting low-loss subspace for personalized federated learning, *in* 'Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining', ACM.

He, K., Zhang, X., Ren, S. and Sun, J. (2016), Deep residual learning for image recognition, *in* '2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)'.

Hoffpauir, K., Simmons, J., Schmidt, N., Pittala, R., Briggs, I., Makani, S. and Jararweh, Y. (2023), 'A survey on edge intelligence and lightweight machine learning support for future applications and services', *J. Data and Information Quality* **15**(2).

Hsu, T.-M. H., Qi, H. and Brown, M. (2019), 'Measuring the effects of non-identical data distribution for federated visual classification'.

Jiang, Y., Wang, S., Valls, V., Ko, B. J., Lee, W.-H., Leung, K. K. and Tassiulas, L. (2023), 'Model Pruning Enables Efficient Federated Learning on Edge Devices', *IEEE Transactions on Neural Networks and Learning Systems* **34**(12), 10374–10386.

Krizhevsky, A. (2012), 'Learning multiple layers of features from tiny images', *University of Toronto* .

Li, Z., Li, H. and Meng, L. (2023), 'Model compression for deep neural networks: A survey', *Computers* **12**(3).

Lin, M., Ji, R., Wang, Y., Zhang, Y., Zhang, B., Tian, Y. and Shao, L. (2020), 'HRank: Filter Pruning using High-Rank Feature Map'. arXiv:2002.10179 [cs].

Lin, R., Xiao, Y., Yang, T.-J., Zhao, D., Xiong, L., Motta, G. and Beaufays, F. (2022), 'Federated Pruning: Improving Neural Network Efficiency with Federated Learning'. arXiv:2209.06359 [cs].

Liu, S., Yu, G., Yin, R., Yuan, J., Shen, L. and Liu, C. (2022), 'Joint Model Pruning and Device Selection for Communication-Efficient Federated Edge Learning', *IEEE Transactions on Communications* **70**(1), 231–244. Conference Name: IEEE Transactions on Communications.

McMahan, H. B., Moore, E., Ramage, D., Hampson, S. and y Arcas, B. A. (2023), 'Communication-efficient learning of deep networks from decentralized data'.

Nikolić, D., Andrić, D. and Nikolić, V. (2023), 'Guided Transfer Learning'. arXiv:2303.16154 [cs].

Olson, R. S. and Moore, J. H. (2016), Tpot: A tree-based pipeline optimization tool for automating machine learning, *in* F. Hutter, L. Kotthoff and J. Vanschoren, eds, 'Proceedings of the Workshop on Automatic Machine Learning', Vol. 64 of *Proceedings of Machine Learning Research*, PMLR, New York, New York, USA, pp. 66–74.

Rahman, K. M. J., Ahmed, F., Akhter, N., Hasan, M., Amin, R., Aziz, K. E., Islam, A. K. M. M., Mukta, M. S. H. and Islam, A. K. M. N. (2021), 'Challenges, applications and design aspects of federated learning: A survey', *IEEE Access* **9**, 124682–124700.

Shlezinger, N., Rini, S. and Eldar, Y. C. (2020), The communication-aware clustered federated learning problem, *in* '2020 IEEE International Symposium on Information Theory (ISIT)'.

Simonyan, K. and Zisserman, A. (2015), 'Very deep convolutional networks for large-scale image recognition'.

Tingting, W., Song, C. and Zeng, P. (2023), 'Efficient federated learning on resource-constrained edge devices based on model pruning', *Complex & Intelligent Systems* **9**.

Wu, T., Song, C. and Zeng, P. (2023), 'Efficient federated learning on resource-constrained edge devices based on model pruning', *Complex & Intelligent Systems* **9**(6), 6999–7013.

You, Z., Yan, K., Ye, J., Ma, M. and Wang, P. (2019), 'Gate Decorator: Global Filter Pruning Method for Accelerating Deep Convolutional Neural Networks'. arXiv:1909.08174 [cs, eess].

Yu, S., Muñoz, J. P. and Jannesari, A. (2023), 'Bridging the gap between foundation models and heterogeneous federated learning'.

Yu, S., Nguyen, P., Anwar, A. and Jannesari, A. (2021), 'Adaptive dynamic pruning for non-iid federated learning', *CoRR* **abs/2106.06921**.
URL: *https://arxiv.org/abs/2106.06921*

Zhou, G., Xu, K., Li, Q., Liu, Y. and Zhao, Y. (2021), 'AdaptCL: Efficient Collaborative Learning with Dynamic and Adaptive Pruning'. arXiv:2106.14126 [cs].

Zhu, H., Xu, J., Liu, S. and Jin, Y. (2021), 'Federated learning on non-iid data: A survey', *Neurocomputing (Amsterdam)* **465**, 371 – 390.

Submission Checklist

1. For all authors...

   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]

   (b) Did you describe the limitations of your work? [Yes]

   (c) Did you discuss any potential negative societal impacts of your work? [Yes]

   (d) Did you read the ethics review guidelines and ensure that your paper conforms to them? `https://2022.automl.cc/ethics-accessibility/` [Yes]

2. If you ran experiments...

   (a) Did you use the same evaluation protocol for all methods being compared (e.g., same benchmarks, data (sub)sets, available resources)? [Yes]

   (b) Did you specify all the necessary details of your evaluation (e.g., data splits, pre-processing, search spaces, hyperparameter tuning)? [Yes]

   (c) Did you repeat your experiments (e.g., across multiple random seeds or splits) to account for the impact of randomness in your methods or data? [Yes]

   (d) Did you report the uncertainty of your results (e.g., the variance across random seeds or splits)? [Yes]

   (e) Did you report the statistical significance of your results? [Yes]

   (f) Did you use tabular or surrogate benchmarks for in-depth evaluations? [N/A] However, we performed our experiments based on publicly available datasets.

   (g) Did you compare performance over time and describe how you selected the maximum duration? [Yes] We based our choices on previouslt published works.

   (h) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]

   (i) Did you run ablation studies to assess the impact of different components of your approach? [Yes]

3. With respect to the code used to obtain your results...

   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results, including all requirements (e.g., `requirements.txt` with explicit versions), random seeds, an instructive `README` with installation, and execution commands (either in the supplemental material or as a URL)? [Yes]

   (b) Did you include a minimal example to replicate results on a small subset of the experiments or on toy data? [Yes]

   (c) Did you ensure sufficient code quality and documentation so that someone else can execute and understand your code? [Yes]

   (d) Did you include the raw results of running your experiments with the given code, data, and instructions? [Yes]

   (e) Did you include the code, additional data, and instructions needed to generate the figures and tables in your paper based on the raw results? [Yes]

4. If you used existing assets (e.g., code, data, models)…

   (a) Did you cite the creators of used assets? [N/A]

   (b) Did you discuss whether and how consent was obtained from people whose data you're using/curating if the license requires it? [N/A]

   (c) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you created/released new assets (e.g., code, data, models)…

   (a) Did you mention the license of the new assets (e.g., as part of your code submission)? [Yes]

   (b) Did you include the new assets either in the supplemental material or as a URL (to, e.g., GitHub or Hugging Face)? [Yes]

6. If you used crowdsourcing or conducted research with human subjects…

   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

7. If you included theoretical results…

   (a) Did you state the full set of assumptions of all theoretical results? [N/A]

   (b) Did you include complete proofs of all theoretical results? [N/A]

A Ablation Study on T_p

We perform an ablation study to assess the sensitivitiy of our method to the pruning threshold parameter T_p. In particular, we check how the average accuracy and loss for the global model predictions vary for $T_p \in \{0.1, 0.2, 0.3, 0.4, 0.5\}$. We do this on two datasets: MINST in Figure 4 and CIFAR10 in Figure 5 from the Pathological Non-IID scenario. In both cases, $T_p = 0.3$ seems the most convenient choice.

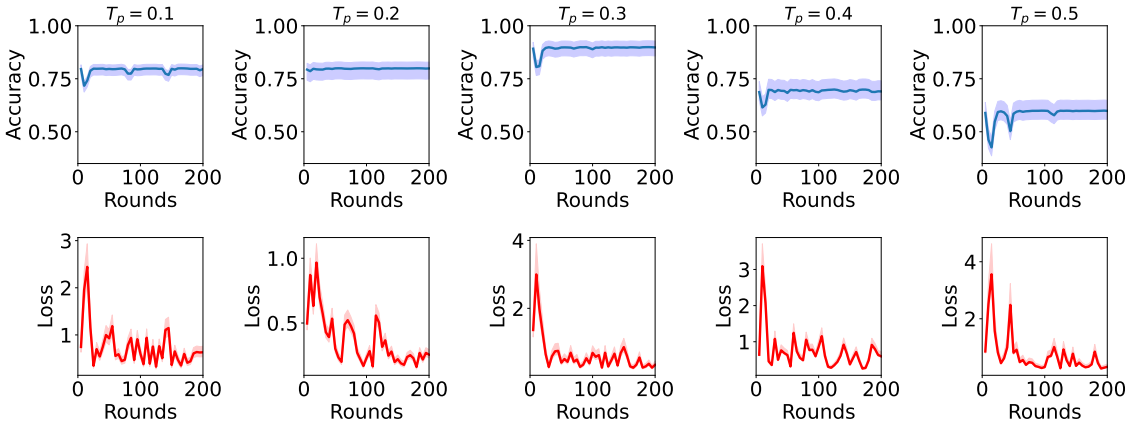

Figure 4: Ablation on T_p for MINST/Non-IID based on average accuracy (top) and loss (bottom).

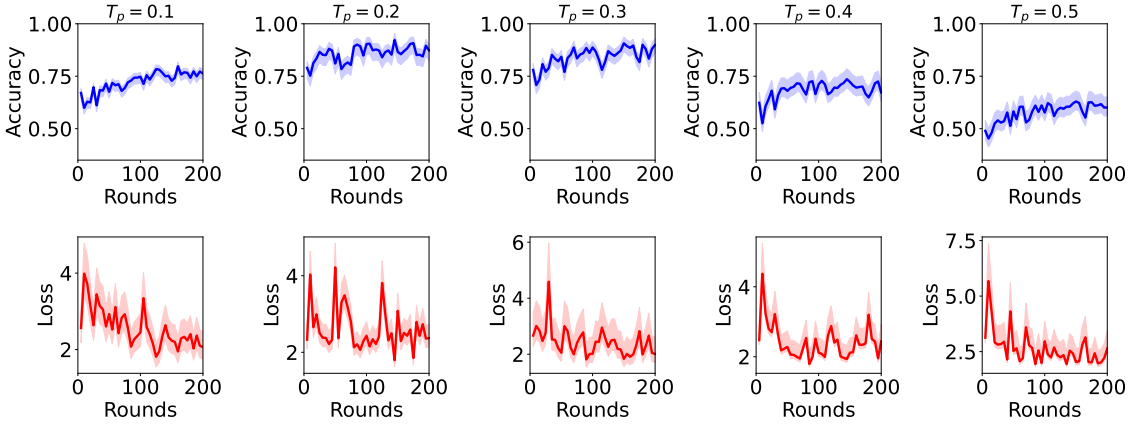

Figure 5: Ablation on T_p for CIFAR10/Non-IID based on average accuracy (top) and loss (bottom).

B Partitioning approaches

Pathological Non-IID. This experimental configuration is delineated by each client possessing data exclusively from two distinct classes within a broader multi-class dataset. Figure 6 illustrates this "pathological" data partitioning scenario within the CIFAR10 dataset across 20 clients. For our experiments, we select the MNIST dataset (Deng, 2012) with a six-layer CNN (7628484 parameters) and the CIFAR10 dataset (Krizhevsky, 2012) with EfficientNet-B3 architecture (10838784 parameters), following the guidelines in (McMahan et al., 2023) and (Tingting et al., 2023).

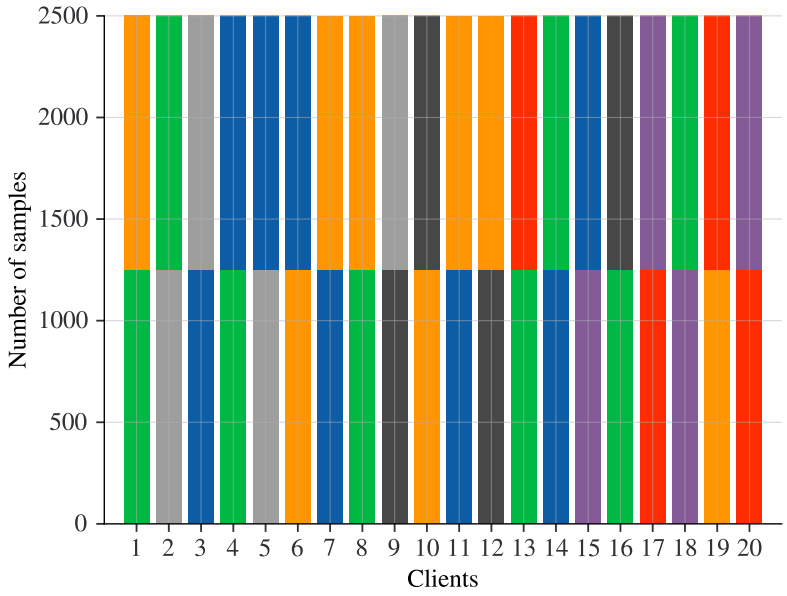

Figure 6: Illustration of pathological data partitioning on CIFAR10 for 20 clients, where each color represents a different class.

Dirichlet-based Non-IID. This advanced experimental setup, as introduced by Hsu et al. (2019), utilizes the Dirichlet distribution, modulated by a concentration parameter α. Let $\boldsymbol{p} = (p_1, p_2, ..., p_N)$ be the class distribution for a given client, where N is the number of classes. The Dirichlet distribution is defined as $\boldsymbol{p} \sim \text{Dir}(\alpha \cdot \mathbf{1}_N)$, where "Dir" denotes the Dirichlet distribution, α is the concentration parameter, and $\mathbf{1}_N$ is a N-dimensional vector of ones. In this context, a low value of α, or $\alpha \to 0$, leads to distributions where most of the probability mass is concentrated on a single class, thereby indicating that each client's data is restricted to a single class. Conversely, as $\alpha \to \infty$, $\boldsymbol{p}$ approaches a uniform distribution, ensuring that the samples are evenly split across all clients. Figure 7 illustrates this "Dirichlet-based Non-IID" data partitioning scenario within the CIFAR100 dataset across 20 clients, with individual colors denoting separate classes.

To address the complexities of larger datasets, we have extended our evaluation to include CIFAR100 (Krizhevsky, 2012) with a $\alpha = 100$, employing ResNet (23755900 parameters) (He et al., 2016) in alignment with the methodology proposed in (Hahn et al., 2022).

LEAF Non-IID. Utilizing the popular LEAF benchmark for FL (Caldas et al., 2019), we selected the FEMNIST and Shakespeare datasets to simulate closer real-world FL scenarios, with each dataset designed for specific tasks. The FEMNIST dataset is defined for a multi-class classification challenge involving 62 distinct classes. Conversely, the Shakespeare dataset is tailored for a next-character prediction task, requiring models to predict the subsequent character from a sequence of 80 characters, thereby testing the model capabilities in sequential data processing and language

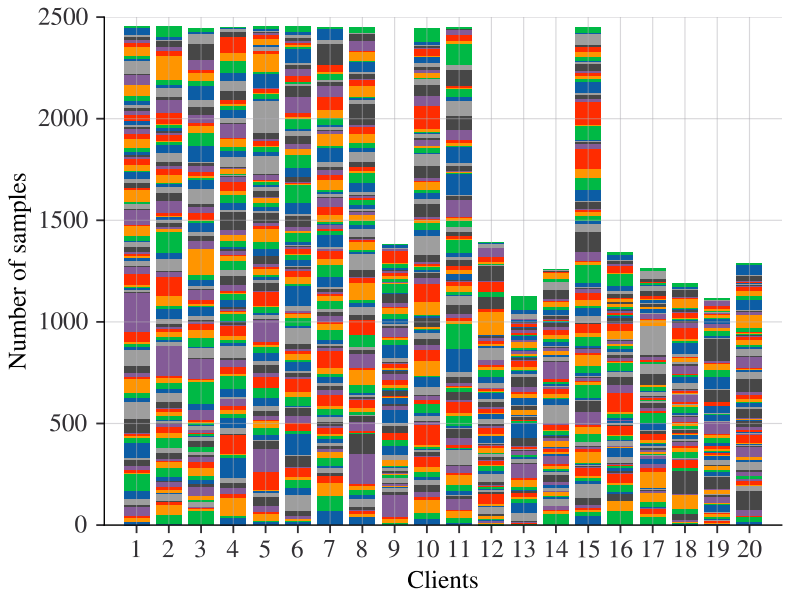

Figure 7: Illustration of Dirichlet-based Non-IID data partitioning on CIFAR100 for 20 clients, where each color represents a different class.

modeling. The incorporation of the next-character prediction task allows for a comprehensive assessment of `AutoFLIP` adaptability and performance across diverse task types and deep neural network architectures, such as Long Short-Term Memory (LSTM) networks.

In our experimental setup, we employed the FEMNIST-CNN architecture, as delineated in (Caldas et al., 2019), for the FEMNIST dataset. For the Shakespeare dataset, we utilized a two-layer (LSTM) (5040000 parameters) model, in accordance with the specifications provided in (McMahan et al., 2023).

C Loss plots

We present in Figure 8 the loss convergence profiles for the global model participating in the FL procedure.for the global model. Here, we compare `AutoFLIP` to the different federated pruning strategies evaluated on both image recognition and text prediction tasks using five distinct datasets: MNIST, CIFAR10, CIFAR100, FEMNIST, and Shakespeare. Due to the varying complexities of each task, we use different model structures for different datasets.

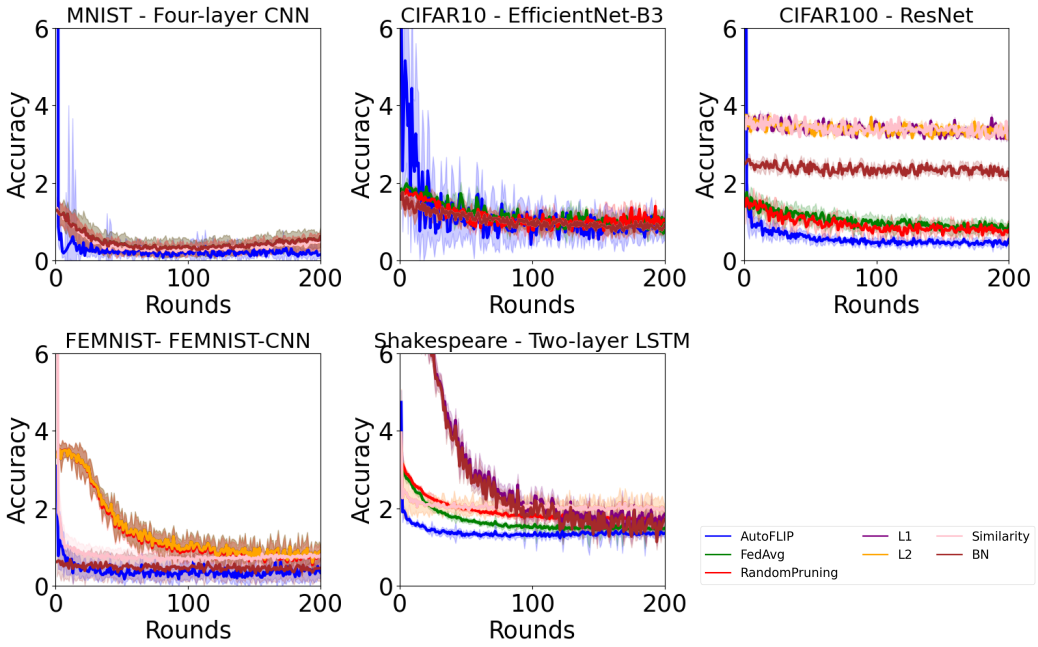

Figure 8: Average loss convergence profiles for the global model within the FL framework.

D Inference acceleration

In this section, we discuss the inference acceleration of AutoFLIP. When performing inference on the client's side with the pruned sub-model, we accelerate the inference time and reduce the computational consumption. Figure 9 shows the inference acceleration comparison after applying AutoFLIP. Notably, the FLOPs (floating point operations per second) in all the evaluated models are reduced. Table 1 shows that the Six-layer CNN deployed for the pathological non-IID experiment with MNIST, experienced a substantial decrease in computational load, equal to a 41.62% reduction in FLOPs. EfficientNet-B3, used for CIFAR10 in the pathological non-IID experiment, saw further improvements, reaching a FLOPs reduction of 46.44%. The deeper ResNet model, designed for CIFAR100 in the Dirichlet-based non-IID experiment, achieved a significant reduction in FLOPs, over 50%, highlighting the potential of AutoFLIP to streamline deep networks for more efficient inference. The FEMNIST-CNN and LSTM models, employed for the LEAF non-IID experiment, showcased a FLOPs reduction equal to 56.49% and 44.44%, respectively.

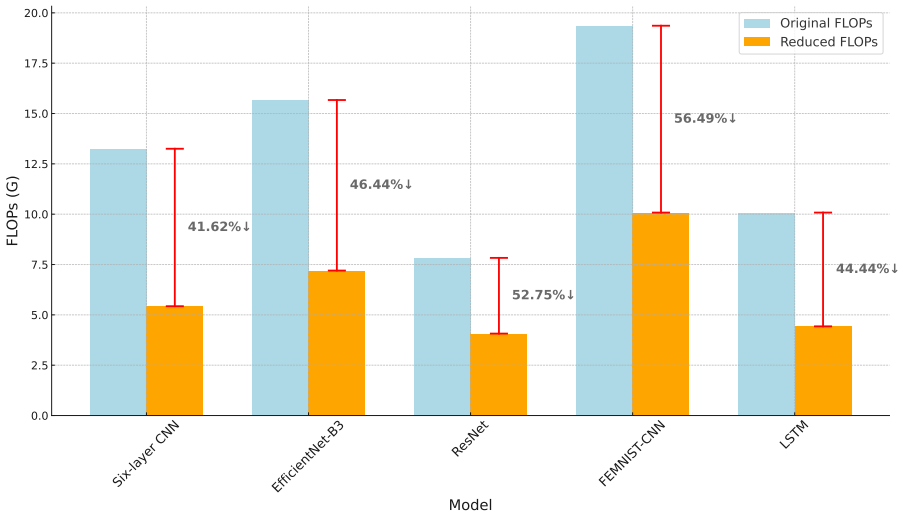

Figure 9: Original FLOPs and reduced FLOPs

Table 1: FLOPs comparison

Model	Compression Rate	Original FLOPs	Reduced FLOPs	FLOPs % Reduced
Six-layer CNN	1.74	13.25 G	5.43 G	41.62% ↓
EfficientNet-B3	2.1	15.67 G	7.20 G	46.44% ↓
ResNet	1.58	7.83 G	4.07 G	52.75% ↓
FEMNIST-CNN	1.8	19.36 G	10.08 G	56.49% ↓
LSTM	1.8	10.08 G	4.43 G	44.44% ↓

E Training efficiency

To ascertain AutoFLIP's impact on enhancing training efficiency within FL frameworks, we delve into an examination of the associated communication costs. For a practical perspective, the deployed models are trained to achieve a 90% accuracy threshold. The cost function employed for this evaluation is defined as:

$$\text{Cost} = \#\,\text{Parameters} \times \#\,\text{Rounds to Reach Target Accuracy} \times \#\,\text{Clients} \times \text{Sample Rate}.$$

In Table 2, we observe the effectiveness of AutoFLIP in reducing communication costs across various non-IID scenarios with different models and datasets. Notably, the Six-layer CNN model, used in the MNIST dataset for the Pathological non-IID experiment, demonstrated a significant reduction in communication costs by 41.61%, which underscores AutoFLIP's effectiveness in simpler architectures. This efficiency extends to more complex architectures, like EfficientNet-B3 and ResNet, employed for the CIFAR10 and CIFAR100 datasets respectively for the Dirichlet-based non-IID experiment, which also saw notable cost reductions of 30.93% and 29.88%. Similarly, the FEMNIST-CNN and LSTM models, used in the LEAF non-IID experiment, exhibited reductions in communication costs by 19.54% and 19.29%, respectively. These results highlight AutoFLIP's broad applicability and substantial impact on training efficiency across a range of model complexities and dataset types.

Table 2: Comparison of the total communication costs

Model	Rounds AutoFLIP	Rounds NoAutoFLIP	Cost AutoFLIP	Cost NoAutoFLIP	% Cost Reduced
Six-layer CNN	3	58	189.45 GB	324.43 GB	41.61% % ↓
EfficientNet-B3	27	39	290.26 GB	420.27 GB	30.93% % ↓
ResNet	7	49	712.70 GB	1016.40 GB	29.88% % ↓
FEMNIST-CNN	280	348	369.06 GB	458.69 GB	19.54% % ↓
LSTM	243	301	122.47 GB	151.74 GB	19.29 % ↓

F Computation Cost

To evaluate AutoFLIP's role in reducing computational effort, we investigate the number of parameters processed for a single client. Distinguishing between computational efforts on the global model and the clients is essential, with a particular focus on the client side. For AutoFLIP, each client handles a substantial number of parameters over an additional 150 exploration epochs (E_{exp}). From a practical standpoint, we compare AutoFLIP and FedAvg with RandomPruning with the same compression rate. The models are trained to meet a 90% of global accuracy. We define the computation cost function as:

$$\text{Computation cost for single client} = \text{Total Parameters Processed} \times \#\,\text{Epochs} \times \text{Sample Rate}$$

In Table 3, the pathological non-IID experiment with MNIST using the Six-layer CNN model shows a significant reduction in computational cost by 62.51%. This efficiency extends to more complex architectures like EfficientNet-B3 and ResNet, used for the CIFAR10 and CIFAR100 datasets respectively, with cost reductions of 46.41% and 58.22%. Similarly, the FEMNIST-CNN and LSTM models, employed in the LEAF non-IID experiment, demonstrated reductions in computational costs by 45.99% and 29.60% respectively. These results underline AutoFLIP's broad applicability and substantial impact on reducing computational efforts across diverse model architectures and dataset types.

Table 3: Comparison of the total computation costs

Model	Processed parameters AutoFLIP	Processed parameters NoAutoFLIP	% Cost Reduced
Six-layer CNN	535,309,170	1,428,653,880	62.51% ↓
EfficientNet-B3	2,005,175,040	3,740,891,680	46.41% ↓
ResNet	3,919,723,500	9,378,097,500	58.22% ↓
FEMNIST-CNN	15,303,653,440	28,338,578,100	45.99% ↓
LSTM	5,871,600,000	8,335,200,000	29.60% ↓