SPATL: Salient Parameter Aggregation and Transfer Learning for Heterogeneous Federated Learning

Sixing Yu\(^1\), Phuong Nguyen\(^1\), Waqwoya Abebe\(^1\), Wei Qian\(^1\), Ali Anwar\(^2\), and Ali Jannesari\(^1\)

\(^1\)Department of Computer Science, Iowa State University, Ames, IA, USA
\(^2\)IBM Almaden Research Center, San Jose, CA, USA
\(^1\)\{yusx, dphoon, wmabebe, wqi, jannesari\}@iastate.edu
\(^2\)ali.anwar2@ibm.com

Abstract—Federated learning (FL) facilitates the training and deploying AI models on edge devices. Preserving user data privacy in FL introduces several challenges, including excessive communication costs, limited resources, and data heterogeneity. In this paper, we propose SPATL, an FL method that addresses these issues by: (a) introducing a salient parameter selection agent and communicating selected parameters only; (b) splitting a model into a shared encoder and a local predictor, and transferring its knowledge to heterogeneous clients via the locally customized predictor. Additionally, we leverage a gradient control mechanism to further speed up model convergence and increase robustness of training processes. Experiments demonstrate that SPATL reduces communication overhead, accelerates model inference, and enables stable training processes with better results compared to state-of-the-art methods. Our approach reduces communication cost by up to 86.45\%, accelerates local inference by reducing up to 39.7\% FLOPs on VGG-11, and requires 7.4× less communication overhead when training ResNet-20. \(^1\)

Index Terms—Federated Learning, Heterogeneous System, Machine Learning, ML, FL

I. INTRODUCTION

Distributed machine learning (ML) is extensively used to solve real-world problems in high performance computing (HPC) environments. Typically, training data is first collected at a central location like a data center or HPC cluster. Afterwards, the data is carefully distributed across the cluster nodes based on the availability of resources. Training is conducted in a distributed manner and a resource and data aware fashion. However, new legislation such as General Data Protection Regulation (GDPR) \(^1\) and Health Insurance Portability and Accountability Act (HIPAA) \(^2\) prohibit user data collection. In response to user privacy concerns, federated learning (FL) \(^3\) was proposed to train ML models while maintaining local data privacy (restricting direct access to private user data).

FL trains a shared model on edge devices (e.g., mobile phones) by aggregating locally trained models on a cloud/central server. This setting, however, presents three key challenges: First, the imbalance/non-independent identically distributed (non-IID) local data easily causes training failure in the decentralized environment. Second, frequent sharing of model weights between edge devices and the server incurs excessive communication overhead. Lastly, the increasing demand of computing, memory, and storage for AI models (e.g., deep neural networks – DNNs) makes it hard to deploy on resource-limited edge devices. These challenges suggest that designing efficient FL models and deploying them effectively will be critical in achieving higher performance on future systems. Recent works in FL and its variants \(^4\)–\(^6\) predominantly focus on learning efficiency, i.e., improving training stability and using the minimum training rounds to reach the target accuracy. However, these solutions further induce extra communication costs. As such, there is no superior solution to address the three key issues jointly. Furthermore, the above methods aim to learn a uniform shared model for all the heterogeneous clients. But this provides no guarantees of model performance on every non-IID local data.

Deep learning models are generally over-parameterized and can easily overfit during local FL updates; in which case, only a subset of salient parameters decide the final prediction outputs. It is therefore unnecessary to aggregate all the parameters of the model. Additionally, existing work \(^7\), \(^8\) demonstrate that a well-trained deep learning model can be easily transferred to non-IID datasets. Therefore, we propose to use transfer learning to address the data heterogeneity issue of Federated Learning. As such, we train a shared model and transfer its knowledge to heterogeneous clients by keeping its output layers customized on each client. For instance, computer vision models (e.g., CNNs) usually consist of an encoder part (embed the input instance) and a predictor head (output layers). In this case, we only share the encoder part in the FL communication process and transfer the encoder’s knowledge to local Non-IID data using a customized local predictor.

Although we use the encoder-predictor based model as an example, our idea can be extend to all AI models whose knowledge is transferable (i.e., we can transfer deep learning model by keeping the output layers heterogeneous on local clients).

Based on these observations, we propose an efficient FL method through Salient Parameter Aggregation and Transfer Learning (SPATL). Specifically, we train the model’s encoder in a distributed manner through federated learning and transfer its knowledge to each heterogeneous client via locally deployed predictor heads. Additionally, we deploy a pre-
trained local salient parameter selection agent to select the encoder’s salient parameters based on its topology. Then, we customize the pre-trained agent on each local client by slightly fine-tuning its weights through online reinforcement learning. We reduce communication overhead by only uploading the selected salient parameters to the aggregating server. Finally, we leverage a gradient control mechanism to correct the encoder’s gradient heterogeneity and guide the gradient towards a generic global direction that suits all clients. This further stabilizes the training process and speeds up model convergence.

In summary, the contributions of SPATL are:

- **SPATL** reduces communication overhead in federated learning by introducing salient parameter selection and aggregation for over-parameterized models. This also results in accelerating the model’s local inference.
- **SPATL** addresses data heterogeneity in federated learning via knowledge transfer of the trained model to heterogeneous clients.
- **SPATL** utilizes a salient parameter selection agent by leveraging online reinforcement learning for fine-tuning.
- **SPATL** enables scalable federated learning to allow large-scale decentralized training.
- We evaluate **SPATL** on a medium-scale, 100 clients setup using Non-IID Benchmark [9]. Our results show that compared to state-of-the-art FL solutions, when optimizing a model, SPATL reduces communication cost by up to 7.4×, improves model performance by up to 19.86%, and reduces inference time by up to 39.7% FLOPs.

II. Motivation – Use Cases

With advancements in the performance of mobile and embedded devices, more and more applications are moving to decentralized learning on the edge. Improved ML models and advanced weight pruning techniques mean a significant amount of future ML workload will come from decentralized training and inference on edge devices [10]. Edge devices operate under strict performance, power, and privacy constraints, which are affected by factors such as model size and accuracy, training and inference time, and privacy requirements. Many edge applications, such as self-driving cars, could not be developed and validated without HPC simulations, in which edge devices. Hence, scalability is important in FL and HPC settings. Lastly, training data on client edge devices depends on the user’s unique usage causing an overall non-IID [3], [23] user dataset. Data heterogeneity is a major problem in decentralized model training [4]–[6], [11], [23]–[29]. Thus designing efficient decentralized learning models and deploying them effectively will be crucial to improve performance of future edge computing and HPC.

III. Related Work

A. Federated Learning

With increasing concerns over user data privacy, federated learning was proposed in [3], to train a shared model in a distributed manner without direct access to private data. The algorithm FedAvg [3] is simple and quite robust in many practical settings. However, the local updates may lead to divergence due to heterogeneity in the network, as demonstrated in previous works [4], [26], [30]. To tackle these issues, numerous variants have been proposed [4]–[6]. For example, FedProx [6] adds a proximal term to the local loss, which helps restrict deviations between the current client model and the global model. FedNova [5] introduces weight modification to avoid gradient biases by normalizing and scaling the local updates. SCAFFOLD [4] corrects update direction by maintaining drift variances, which are used to estimate the overall update direction of the server model. Nevertheless, these variants incur extra communication overhead to maintain stable training. Notably, in FedNova and SCAFFOLD, the average communication cost in each communication round is approximately 2× compared to FedAvg.

Numerous research papers have addressed data heterogeneity (i.e. non-IID data among local clients) in FL [23], [25], [27], [31]–[35], such as adjusting classifier [36], improving client sampling fairness [24], adapting optimization [28], [29], [37], [38], correcting local updates [4], [5], [39], using a tiering mechanism to synchronously update local model parameters within tiers and asynchronously update the global model [40], generating client models from a central hypernetwork model [41], dynamically measuring local model divergence and adaptively adjusting to optimize hyperparameters [42], and using data-free knowledge distillation approach to address heterogeneous FL [43].

Further more, federated learning has been extended in real life applications [44], [45]. One promising solution is personalized federated learning [46]–[51], which tries to learn personalized local models among clients to address data heterogeneity. These works, however, fail to address the extra communication overhead. However, very few works, such as [52], [53] focus on addressing communication overhead in FL. They either use knowledge distillation, or aggregation
Additionally, benchmark federated learning settings have been introduced to better evaluate the FL algorithms. FL benchmark LEAF [54] provides benchmark settings for learning in FL, with applications including federated learning, multi-task learning, meta-learning, and on-device learning. Non-IID benchmark [9] is an experimental benchmark that provides us with Non-IID splitting of CIFAR-10 and standard implementation of SOTAs. Framework Flower [55] provides FL SOTA baselines and is a collection of organized scripts used to reproduce results from well-known publications or benchmarks. IBM Federated Learning [56] provides a basic fabric for FL on which advanced features can be added. It is not dependent on any specific machine learning framework and supports different learning topologies, e.g., a shared aggregator and protocols. It is meant to provide a solid basis for federated learning that enables a large variety of federated learning models, topologies, learning models, etc., particularly in enterprise and hybrid-Cloud settings.

B. Salient Parameter Selection

Since modern AI models are typically over-parameterized, only a subset of parameters determine practical performance. Several network pruning methods have been proposed to address this issue. These methods have achieved outstanding results and are proven techniques to drastically shrink model sizes. However, traditional pruning methods [57]–[59] require time-consuming re-training and re-evaluating to produce a potential salient parameter selection policy. Recently, AutoML pruning algorithms [60], [61] offered state-of-the-art (SoTA) results with higher versatility. In particular, reinforcement learning (RL)-based methods [62]–[65], which model the neural network as graphs and use GNN-based RL agent to search for pruning policy present impressive results. However, AutoML methods need costly computation to train a smart agent, which is impractical to deploy on resource-limited edge FL devices.

The enormous computational cost and effort of network pruning makes it difficult to directly apply in federated learning. To overcome challenges of previous salient parameter selection methods and inspired by the RL-based AutoML pruning methods, we utilize a salient parameter selection RL agent pre-trained on the network pruning task. Then with minimal fine-tuning, we implemented an efficient salient parameter selector with negligible computational burden.

IV. METHODOLOGY

SPATL consists of three main components: knowledge transfer learning, salient parameter selection agent, and gradient control federated learning. Figure 1 shows the SPATL overview. Unlike mainstream FL solutions, which attempt to train the entire deep learning model, SPATL only trains the encoder part of the model in a distributed manner and transfers the knowledge to heterogeneous clients. In each round of federated learning, the client first downloads the encoder from the cloud aggregator (❼ in Figure 1) and transfers its knowledge using a local predictor through local updates (➋ in Figure 1). After local updates, the salient parameter selection agent will evaluate the training results of the current model based on the model performance (⢀ in Figure 1), and finally selected clients send the salient parameters to the server (⣔ in Figure 1). Additionally, both clients and the server maintain a gradient control variate to correct the heterogeneous gradients, in order to stabilize and smoothen the training process.

A. Heterogeneous Knowledge Transfer Learning

Inspired by transfer learning [7], SPATL aims to train an encoder in FL setting and address the heterogeneity issue through transferring the encoder’s knowledge to heterogeneous clients. Formally, we formulate our deep learning model as an encoder $E(w_{e}, x)$ and a predictor $P(w_{p}, e)$, where $w_{e}$ and $w_{p}$ are encoder and predictor parameters respectively, $x$ is an input instance to the encoder and $e$ is an input instance to the predictor (or embedding).

SPATL shares the encoder $E(w_{e}, x)$ with the cloud aggregator, while the predictor $P^{k}(w_{p}^{k}, e)$ for the $k^{th}$ client is kept private on the client. The forward propagation of the model in the local client $k$ is formulated as follows:

\begin{align}
    e &= E(w_{e}, x), \\
    \hat{y} &= P^{k}(w_{p}^{k}, e)
\end{align}

During local updates, the selected $k^{th}$ client first downloads the shared encoder parameter, $w_{e}$, from the cloud server and optimizes it with the local predictor head, $w_{p}^{k}$, through back propagation. Equation 3 shows the optimization function.

\begin{equation}
    \min_{w_{e},w_{p}^{k}} \mathcal{L}(w_{e},w_{p}^{k}) = \frac{1}{n_{i}} \sum l(w_{e},w_{p}^{k}, x_i, y_i)
\end{equation}

Here, $l$ refers to the loss when fitting the label $y_i$ for data $x_i$, and $n_{i}$ is the constant coefficient.

In federated learning, not all clients are involved in communication during each round. In fact, there is a possibility a client might never be selected for any communication round.
Before deploying the trained encoder on such a client, the client will download the encoder from the aggregator and apply local updates to its local predictor only. After that, both encoder and predictor can be used for that client. Equation 4 shows the optimization function.

$$
\min_{w_p^k} \mathcal{L}(w_p^k) = \frac{1}{n_i} \sum l(w_e, w_p^k, x_i, y_i)
$$  \hspace{1cm} (4)

### B. RL-based Topology-Aware Salient Parameter Selection

One key issue of FL is the high communication overhead caused by the frequent sharing of parameters between clients and the cloud aggregator server. Additionally, we observed that deep learning models (e.g., VGG [66] and ResNet [67]) are usually bulky and over-parameterized. As such, only a subset of salient parameters decide the final output. Therefore, in order to reduce the communication cost, we implemented a local salient parameter selection agent for selecting salient parameters for communication.

Figure 2 shows the idea of a salient parameter agent. Specifically, inspired by topology-aware network pruning task [62], [65], we model the neural network (NN) as a simplified computational graph and use it to represent the NN’s states. Since NNs are essentially computational graphs, their parameters and operations correspond to nodes and edges of the computational graph. We then introduced the graph neural network (GNN)-based reinforcement learning (RL) agent, which takes the graph as input (RL’s environment states) and produces a parameter selection policy from the topology through GNN embedding. Additionally, the RL agent uses the selected sub-model’s accuracy as reward to guide its search for the optimal pruning policy. Training a smart agent directly through RL, however, is costly and impractical to deploy on the edge. To address this issue, we first pre-train the salient parameter agent in the network pruning task, and then customize the pre-trained agent on each local client by slightly fine-tuning its weights through online reinforcement learning (detailed hyper-parameter setting in section V).

#### 1) Reinforcement Learning Task Definition: Defining environment states, action space, reward function, and RL policy

In order to reduce the communication cost, we implemented a local salient parameter selection agent for selecting salient parameters for communication. The parameter selection agent selects a salient parameter and uses the selected sub-model’s accuracy to search for the desired policy.

**Algorithm 1: Salient parameter selection with PPO**

**Input:** shared encoder \( E(w_e) \) and model size constraints \( d \)

**Output:** selected salient parameter and corresponding index

for search step \( k = 1, 2, \ldots \) do

\[ \hat{E}(\hat{w}_e) \leftarrow E(w_e) \]

\[ G \leftarrow \text{computational graph of } \hat{E}(\hat{w}_e) \]

while \( \text{Size}(\hat{E}) > d \) do

\[ a = \text{PPO.Actor_Critic}(G) \]

\[ \text{PPO.Memory}(G, a) \]

\[ \hat{E}(\hat{w}_e) \leftarrow \text{pram_select}(\hat{E}(\hat{w}_e), a) \]

\[ G \leftarrow \text{computational graph of } \hat{E} \]

\[ \text{Reward} = \text{Accuracy}(\hat{E}) \]

if is best Reward then

\[ w \leftarrow \hat{w}_e \]

\[ idx \leftarrow \text{extract the index of } \hat{w}_e \]

\[ \text{PPO.update}(\text{Reward}) \]

Return \( w, idx \)

---

**Fig. 2:** Parameter selection with reinforcement learning. The parameter selection agent selects a salient parameter and uses the selected sub-model’s accuracy to search for the desired policy.

**Fig. 3:** Aggregation with salient parameter. For each client, the aggregator will aggregate partial parameters corresponding to the client’s salient parameter.
Relu, etc.).

**Action Space.** The actions are the sparsity ratios for encoder’s hidden layers. The action space is defined as $a \in [0, 1]^N$, where $N$ is the number of encoder’s hidden layers. The actor network in the RL agent projects the NN’s computational graph to an action vector, as shown in equations 5 and 6.

$$g = \text{GraphEncoder}(G),$$

$$a = \text{mlp}(g)$$

Here, $G$ is the environment state, $g$ is the graph representation, and MLP is a multi-layer perceptron neural network. The graph encoder learns the topology embedding, and the MLP projects the embedding into hidden layers’ sparsity ratios.

**Reward Function.** The reward function is the accuracy $\times 100$ of selected sub-network on validation dataset.

$$\text{Reward} = \text{Accuracy} \times 100$$

2) **Policy Updating:** The RL agent is updated end-to-end through the PPO algorithm. The RL agent trains on the local clients through continual online-learning over each FL round. Equation 8 shows the objective function we used for the PPO update policy.

$$L(\theta) = \mathbb{E}_t[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon)\hat{A}_t)]$$

Here, $\theta$ is the policy parameter (the actor-critic network’s parameter), $\mathbb{E}_t$ denotes the empirical expectation over time steps, $r_t(\theta)$ is the ratio of the probability under the new and old policies, respectively, $\hat{A}_t$ is the estimated advantage at time $t$, and $\epsilon$ is a clip hyper-parameter, usually 0.1 or 0.2.

C. **Generic Parameter Gradient Controlled Federated Learning**

Inspired by stochastic controlled averaging federated learning [4], we propose a generic parameter gradient controlled federated learning to correct the heterogeneous gradient. Due to client heterogeneity, local gradient update directions will move towards local optima and may diverge across all clients. To correct overall gradient divergence by estimating gradient update directions, we maintain control variates both on clients and the cloud aggregator. However, controlling the entire model’s gradients will hurt the local model’s performance on non-IID data. In order to compensate for performance loss, SPATL only corrects the generic parameter’s gradients (i.e., the encoder’s gradients) while maintaining a heterogeneous predictor. Specifically in equation 9, during local updates of the encoder, we correct gradient drift by adding the estimate gradient difference $(c_g - c_l)$.

$$w_e \leftarrow w_e - \eta \nabla(L(w_e, w_p; b) + c_g - c_l)$$

Here, control variate $c_g$ is the estimate of the global gradient direction maintained on the server side, and $c_l$ is the estimate of the update direction for local heterogeneous data maintained on each client. In each round of communication, the $c_l$ is updated as equation 10:

$$c_l^{t+1} \leftarrow c_l - c_g + \frac{1}{E\eta}(w_g - w_e)$$

Here, $E$ is the number of local epochs, and $\eta$ is the local learning rate, while $c_g$ is updated by equation 11:

$$c_g \leftarrow c_g + \frac{1}{|N|} \sum_{k \in K} \Delta c^k$$

Here, $\Delta c^k$ is the difference between new and old local control variates $c_l$ of client $k$, $N$ is the set of clients, and $K$ is the set of selected clients.

Algorithm 2 shows SPATL with gradient controlled FL. In each update round, the client downloads the global encoder’s parameter $w_g$ and update direction $c_g$ from server, and performs local updates. When updating the local encoder parameter $w_e$, $(c_g - c_l)$ is applied to correct the gradient drift. The predictor head’s gradient remains heterogeneous. Before uploading, the local control variate $c_l$ is updated by estimating the gradient drift.

**Algorithm 2: SPATL with gradient controlled FL**

**Server executes:**

initialize $w_g, c_g$.

**for each round** $t = 1, 2, \ldots, T$

**for each client** $k \in K$ in parallel do

communicate with client $k$

$$w^k, i^k, \Delta c^k \leftarrow \text{ClientUpdate}(w_g, c_g)$$

$c_g \leftarrow c_g + \frac{1}{|N|} \sum_{k \in K} \Delta c^k$

**end for**

**for epoch** $1, 2, \ldots, E$

**for each** $E\eta$

**for each batch** $b \in B$

$$w_e \leftarrow w_e - \eta \nabla(L(w_e, w_p; b) - c_l + c_g)$$

$$w_p \leftarrow w_p - \eta \nabla(L(w_e, w_p; b))$$

$c_l' \leftarrow c_l - c_g + \frac{1}{E\eta}(w_g - w_e)$

$$\Delta c \leftarrow (c_l' - c_l)$$

$w_e, idx \leftarrow \text{SalientParameterSelection}(E(w_e), d)$

communicate $w_e, idx, \Delta c$

$c_l \leftarrow c_l'$

1) **Aggregation with Salient Parameters:** Due to the non-IID local training data in heterogeneous clients, salient parameter selection policy varies among the heterogeneous clients after local updates. Since the selected salient parameters have different matrix sizes and/or dimensions, directly aggregating them will cause a matrix dimension mismatch. To prevent this, as Figure 3 shows, we only aggregate partial parameters according to the current client’s salient parameter index on the
server side. Equation 12 shows the mathematical representation of this process.

\[ w_g \leftarrow w_g + \eta \left( \frac{1}{|K|} \sum_{k \in K} (w_g[i^k, :, :] - w^k) \right) \]  \hspace{1cm} (12)

Here, \( w_g \) is the global parameter, \( w_k \) is the \( i^{th} \) client’s salient parameter, \( i^k \) is the \( w^k \)’s index corresponding to the original weights, and \( \eta \) is the update step size. By only aggregating the salient parameter \( w^k \) and its corresponding index \( i^k \) (negligible burdens), we can significantly reduce the communication overhead and avoid matrix dimension mismatches.

V. EXPERIMENT

We conducted extensive experiments to examine SPATL’s performance. Overall, we divided our experiments into three categories: learning efficiency, communication cost, and inference acceleration. We also performed an ablation study and compared SPATL with state-of-the-art FL algorithms.

A. Implementation and Hyper-parameter Setting

Datasets and Models. The experiments are conducted with FEMNIST [54] and CIFAR-10 [69]. In FEMNIST, we follow the LEAF benchmark federated learning setting [54]. In CIFAR-10, we use the Non-IID benchmark federated learning setting [9]. Each client is allocated a proportion of the samples of each label according to Dirichlet distribution (with concentration \( \alpha \)). Specifically, we sample \( p_k \sim Dir_X(\alpha) \) and allocate a \( p_{k,j} \) proportion of the instances to client \( j \). Here we choose the \( \alpha = 0.1 \). The deep learning models we use in the experiment are VGG-11 [66] and ResNet-20/32 [67].

Federated Learning Setting. We follow the Non-IID benchmark federated learning setting and implementation [9]. In SPATL, the models in each client are different. Thus, we evaluate the average performance of models in heterogeneous clients. We experiment on different clients and sample ratio (percentage of participating clients in each round) setting, from 10 clients to 100 clients, and the sample ratio from 0.4 to 1. During local updates, each client updates 10 rounds locally.
The detailed setting can be found in supplementary materials.

**RL Agent Settings.** The RL agent is pre-trained on ResNet-56 by a network pruning task. Fine-tuning the RL agent in the first 10 communication rounds with 20 epochs in each updating round. We only update the MLP’s (i.e., output layers of RL policy network) parameter when fine-tuning. We use the PPO [68] RL policy, the discount factor is $\gamma = 0.99$, the clip parameter is 0.2, and the standard deviation of actions is 0.5. Adam optimizer is applied to update the RL agent, where the learning rate is $3 \times 10^{-4}$ and the $\beta = (0.9, 0.999)$.

**Baseline.** We compare SPATL with the state-of-the-art FL algorithms, such as FedNova [5], FedAvg [3], FedProx [6], and SCAFFOLD [4].

**Experimental Setup.** Our experimental setup ranges from 10 clients to 100 clients, and our experimental scale is on par with existing state-of-the-art works. To better compare and fully investigate the optimization ability, some of our experiments (e.g., communication efficiency) scales are set to be larger than many SOTA method experiments (such as FedNova [5], FedAvg [3], FedProx [6], and SCAFFOLD [4]). Recent FL works, such as FedAT [40], pFedHN [41], QuPeD [51], FedEMA [42], and FedGen [43], their evaluations are within the same scale as our experiments. Since the FL is an optimization algorithm, we mainly investigate the training stability and robustness. The larger experiment scale will show a similar trend.

**FL Benchmark.** We use two standard FL benchmark settings: LEAF [54] and Non-IID benchmark [9]. LEAF [54] provides benchmark settings for learning in FL, with applications including federated learning, multi-task learning, meta-learning, and on-device learning. We use the LEAF to split the FEMNIST into Non-IID distributions. Non-IID benchmark [9] is an experimental benchmark that provides us with Non-IID splitting of CIFAR-10 and standard implementation of SOTAs. Our implementation of FedAvg, FedProx, SCAFFOLD, and FedNova are based on the Non-IID benchmark.

**B. Learning Efficiency**

In this section, we evaluate the learning efficiency of SPATL by investigating the relationship between communication rounds and the average accuracy of the model. Since SPATL learns a shared encoder, each local client has a heterogeneous predictor, and the model’s performance is different among clients. Instead of evaluating a global test accuracy on the server side, we allocate each client a local non-IID training dataset and a validation dataset to evaluate the top-1 accuracy, i.e., the highest probability prediction must be exactly the expected answer, of the model among heterogeneous clients. We train VGG-11 [66] and ResNet-20/32 [67] on CIFAR-10 [69], and 2-layer CNN on FEMNIST [54] separately until the models converge. We then compare model performance results of SPATL with state-of-the-arts (SoTAs) (i.e., FedNova [5], FedAvg [3], FedProx [6], and SCAFFOLD [4]).

Figure 4 experiments show 10 clients setting where we sample all 10 clients for aggregation. The effect of heterogeneity is not significant compared to a real-world scale. SPATL moderately outperforms the SoTAs on CIFAR-10. Results on the 2-layer CNN model trained on FEMNIST however, is an exception; in this case the model trained by SPATL has a slightly lower accuracy than SoTAs. We suspect that it has to do with the small size of the 2-layer CNN and the large data quantity. Particularly, in this case, our “model over-parameterization” assumption no longer holds, making it hard for the salient parameter selection to fit the training data. To verify our analysis, we increase the complexity of our experiments and conduct further experiments on larger scale FL settings. We increase the number of clients to 30, 50, and 100 with different sample ratios.

As heterogeneity rises with the increase in number of clients, SPATL demonstrates superiority in coping with data heterogeneity. Experiment results in Figure 4 show that for more complex FL settings, SPATL outperforms SoTAs with larger margins. In the 30 clients FL setting, for ResNet-20, ResNet-32, and VGG-11, SPATL outperforms the SoTA FL methods. Notably, SPATL yields a better convergence accuracy and a substantially more stable training process. In the 50 clients and 100 clients settings, the experiment improvements become more significant, as SPATL outperforms the SoTAs by a larger margin. Moreover, we noticed that the gradient control based method SCAFFOLD [4] suffers from gradient explosion issues when the number of clients increases. Even though we set a tiny learning rate, the explosion problem persists. Other researchers are facing the same issues when reproducing SCAFFOLD, and our results satisfy finding 6 in [9].

Intuitively, we investigate the model’s accuracy overhead. Figure 5 shows the converge accuracy comparison with SoTAs. SPATL surpasses SoTAs in all the FL settings and achieves higher converge accuracy than SoTAs. Again, the superiority of SPATL grows progressively with the heterogeneity of FL settings. For instance, in ResNet-20 with 30 clients, SPATL outperforms SoTAs only in terms of final converge accuracy. However, when we increase to 50 heterogeneous clients, SPATL achieves 42.54% final accuracy, that is around 10% higher than FedAvg and FedProx (they achieve 32.71% and 32.43% accuracy, respectively). Additionally, it is worth mentioning that, in the 100 clients experiment setting, we compare the accuracy within 200 rounds since all the baselines diverge in 200 rounds except SPATL. This further demonstrates that SPATL optimizes and improves the quality of the model progressively and stably.

Trained model performance on heterogeneous local clients is also an essential indicator when evaluating FL algorithms with regards to deploying AI models on the edge. Since edge devices have various application scenarios and heterogeneous input data, models will likely exhibit divergence on such devices. We further evaluate the robustness and feasibility of FL methods on distributed AI by testing local model accuracies on all clients. Figure 6 shows ResNet-20 on each client’s

\footnote{In Fig. 4 SCAFFOLD [4] diverges with gradient explosion in Non-IID benchmark settings [9] when there are more than 10 clients.}
accuracy after the training is complete for CIFAR-10 (total 10 clients trained by SPATL and SCAFFOLD with 100 rounds). The model trained by SPATL produces better performance across all clients. In particular, the edge model trained by SPATL produces more stable performance among each client, whereas models trained by baselines exhibit more variance. For instance, all the edge models trained by SPATL have similar accuracy performance. Since SPATL uses heterogeneous predictors to transfer the encoder’s knowledge, the model is more robust when dealing with non-IID data. However, our baseline methods (such as SCAFFOLD [4]) share the entire model when training on non-IID clients, leading to a variance in model performance on non-IID clients and causing poor performance on some clients.

C. Communication Efficiency

A key contribution of SPATL that makes it stand out among SoTAs is its significant reduction of communication overhead due to salient parameter selection. Although SoTAs, like FedNova [5] and SCAFFOLD [4], achieve stable training via gradient control or gradient normalization variants, their average communication cost doubles compared to FedAvg [3] as a result of sharing the extra gradient information. We present two experiment settings to evaluate model communication efficiencies. First, we trained all models to a target accuracy and calculated communication cost. Second, we trained all models to converge and calculated the communication cost of each FL algorithm. The communication cost is calculated as:

\[
\text{#Rounds} \times \text{Client’s round cost} \times \text{#Sampled Clients} \quad (13)
\]

Table I shows the detailed information of communication cost (FedAvg [3] as benchmark) when we train models to a target accuracy. SPATL remarkably outperforms SoTAs. In ResNet-20, SPATL reduced communication cost by up to 7.6× (FedNova costs 8.12GB while SPATL only costs 1.1GB). Moreover, SPATL reduced communication by up to 102GB when training VGG-11 compared to FedNova.

There are two main benefits of SPATL in reducing communication overhead. First, training in SPATL is much more efficient and stable. Our experiments show that SPATL requires fewer communication rounds to achieve target accuracy. For example, in VGG-11, FedProx uses 296 training rounds while SPATL uses 250 less rounds to achieve the same target accuracy. This significantly reduces the communication cost. We provide a more comprehensive comparison to show the number of rounds different models take to achieve target accuracy. As Figure 7 shows, we try different FL settings, and SPATL consistently requires fewer rounds than SoTAs in most of the FL settings (except in ResNet-20 with 10 clients and 80% target accuracy, SPATL requires 3 rounds more than SCAFFOLD). However, as shown in Table I), the total communication cost of SPATL is significant less than all others. Second, since the salient parameter selection agent selectively uploads partial parameters, SPATL significantly reduces communication cost. As shown in Table I, compared to the gradient control based methods, such as SCAFFOLD [4] and FedNova [5], SPATL remarkably reduces per round communication cost. For instance, SPATL uses 2× less round costs in ResNet-20 compared to the traditional FL, such as FedAvg [3]. Even when factoring in gradient control information, salient parameter selection enables SPATL to drop unnecessary communication burdens, which makes its round costs comparable to FedAvg.

Furthermore, we investigated the convergence accuracy of models using SoTAs and SPATL optimizations. We consider a performance upper bound by creating a hypothetical centralized case where images are heterogeneously distributed across 30, 50, and 100 clients. Table II shows the results of training the models to convergence. Compared to FedAvg [3], gradient control based FL algorithms have higher accuracy at the expense of communication efficiency. For instance,
FedNova [5] achieves slightly higher accuracy. However, their communication budget increases by more than 2×. Models optimized by SPATL achieve significantly higher accuracy than all other baselines. Especially on VGG-11 with 50 clients, SPATL achieves 17.8%, 19.86%, and 17.62% higher accuracy than FedAvg, FedNova, and FedProx respectively. Particularly, unlike FedNova, which sacrifices communication cost for higher accuracy, SPATL takes advantage of salient parameter selection to achieve higher accuracy with relatively negligible communication overhead. For instance, when training the ResNet-20 for 30 and 50 clients, SPATL achieves the best accuracy with the lowest communication cost. The results further show that SPATL remarkably reduces the communication cost and has a higher capacity to train models for better performance.

D. Inference Acceleration

In this section, we evaluate the inference acceleration of SPATL. Local inference is a crucial measure for deploying AI models on the edge since edge devices have limited computing power, and edge applications (e.g., self-driving car) are inference sensitive. In SPATL, when the salient parameter selection agent selects salient parameters, it prunes the model as well. For a fair evaluation of inference, instead of recording the actual run time (run time may vary on different platforms) of pruned models, we calculated the FLOPs (floating point operations per second).

Table III shows the inference acceleration status after training is complete. SPATL notably reduced the FLOPs in all the evaluated models. For instance, in ResNet-32, the average FLOPs reduction among 10 clients is 29.6%, and the client with the highest FLOPs reduction achieves 38.4% fewer FLOPs than the original model, while the client models have a relatively low sparsity ratio (the sparsity ratio represents the ratio of salient parameters compared to the entire model parameters). The low sparsity ratio can further benefit by accelerating the models on parallel platforms, such as GPUs. Additionally, we evaluate the salient parameter selection agents’ pruning ability and compare it with SoTA pruning methods. As shown in Table V, our agent achieves outstanding results in pruning task and outperforms popular AutoML pruning baselines. The results indicate that SPATL can significantly accelerate model inference with acceptably small accuracy loss.

E. Transferbility of Learned Model

In SPATL, since only the partial model (i.e., knowledge encoder) is trained in a distributed manner, we conducted a transferability comparison experiment to test for successful transfer of knowledge among heterogeneous edge clients. Specifically, we transfer the neural network trained by SPATL and SoTAs (e.g., FedAvg, FedNova, SCAFFOLD, etc.) separately to a new portion of data and compare the performance of transferred models. The experimental settings are as follows: we split the CIFAR-10 [69] into two separate datasets, one with 50K images (for federated learning) and another with 10K images (for transfer learning after FL is finished). We use ResNet-20 and set 10 clients for federated learning, where each client has 4k image local training data and 1k validation set. Transfer learning was conducted in a regular manner without involving training distribution.

Table IV shows the results. The model trained by SPATL achieves comparable transfer learning results with the SoTAs. This further shows that SPATL, which only trains a shared encoder in a distributed manner (i.e., as opposed to training the entire model), can successfully learn and transfer the knowledge of a heterogeneous dataset.

F. Ablation Study

1) Salient Parameter Selection vs. No Parameter Selection: Modern AI models are huge (involving billions of parameters) and often over-parameterized. Thus, only a subset of salient parameters can significantly affect the model’s final performance. As such, a reasonable pruning of redundant parameters might not negatively impact model training. This section investigates the impact of salient parameter selection on federated learning. Specifically, we compare SPATL with and without salient parameter selection. Figure 8 shows the re-
TABLE II: Comparison of communication cost with SoTAs to achieve model convergence.

| Method    | Model   | Clients | Sample Ratio | Converge Rounds | Communication Cost | Avg. Converge Acc. | Δ Acc. |
|-----------|---------|---------|--------------|-----------------|-------------------|--------------------|--------|
|           |         |         |              |                 | Round/Client Total | Speedup            |        |
|           |         |         |              |                 |                   |                    |        |
|           | FedAvg [3] | ResNet-20 | 30           | 0.4             | 170               | 2.1MB              | 4.18GB | 60.58% | 0%      |
|           | FedAvg [3] | ResNet-32 | 30           | 0.4             | 140               | 3.2MB              | 5.25GB | 64.92% | 0%      |
|           | FedAvg [3] | VGG-11   | 50           | 0.7             | 171               | 42MB               | 246GB  | 32.68% | 0%      |
|           | FedNova [5] | ResNet-20 | 50           | 0.7             | 200               | 4.2MB              | 7.83GB | 32.71% | 0%      |
|           | FedNova [5] | ResNet-32 | 50           | 0.7             | 149               | 6.4MB              | 11.18GB| 38.35% | +5.64%  |
|           | FedNova [5] | VGG-11   | 100          | 0.4             | 287               | 84MB               | 941GB  | 29.90% | -2.78%  |
|           | FedProx [6] | ResNet-20 | 30           | 0.4             | 157               | 2.1MB              | 3.86GB | 64.50% | +3.92%  |
|           | FedProx [6] | ResNet-32 | 50           | 0.7             | 200               | 4.2MB              | 13.85GB| 38.35% | -0.28%  |
|           | FedProx [6] | VGG-11   | 100          | 0.4             | 195               | 6.4MB              | 11.18GB| 67.48% | +2.56%  |
|           | SCAFFOLD [4] | ResNet-20 | 50           | 0.7             | 200               | 4.2MB              | 13.85GB| 38.35% | -0.28%  |
|           | SCAFFOLD [4] | ResNet-32 | 50           | 0.7             | 149               | 6.4MB              | 11.18GB| 67.48% | +2.56%  |
|           | SCAFFOLD [4] | VGG-11   | 100          | 0.4             | 195               | 6.4MB              | 11.18GB| 67.48% | +2.56%  |
|           | SPATL (Ours) | ResNet-20 | 30           | 0.4             | 140               | 2.1MB              | 3.86GB | 64.50% | +3.92%  |
|           | SPATL (Ours) | ResNet-32 | 50           | 0.7             | 171               | 4.2MB              | 7.83GB | 32.71% | 0%      |
|           | SPATL (Ours) | VGG-11   | 100          | 0.4             | 156               | 6.4MB              | 11.18GB| 67.48% | +2.56%  |

TABLE III: Inference acceleration.

| Model   | Avg. sparsity ratio | Avg. FLOPs ↓ | Highest FLOPs ↓ |
|---------|---------------------|--------------|-----------------|
| VGG-11  | 70%                 | 29.9%        | 39.7%           |
| ResNet-20 | 84%                | 25.8%        | 31.9%           |
| ResNet-32 | 82%                | 29.6%        | 38.4%           |

TABLE IV: Transferability of model trained by FL.

| Method     | Model   | Org. Acc. | Transferred Acc. |
|------------|---------|-----------|------------------|
| FedAvg [3] | ResNet-20 | 78.5      | 92.16            |
| FedAvg [3] | ResNet-32 | 76.9      | 93.66            |
| FedProx [6] | ResNet-20 | 76.8      | 93.03            |
| SCAFFOLD [4] | ResNet-20 | 81.8      | 91.7             |
| SPATL (Ours) |          | 84.3      | 93.83            |

Fig. 8: Parameter selection vs. no parameter selection.

Fig. 9: (a) Transfer learning vs. no transfer learning. (b) Gradient control vs. no gradient control.

The results. We conducted the experiment on ResNet-20 with various FL settings. All of the results indicate that properly pruning some unimportant weights of over-parameterized networks will not harm training stability in federated learning. Instead, it might produce better results in some cases. Especially in the 10 clients setting, SPATL optimized a higher quality model after applying the parameter selection. We further evaluate the salient parameter agent by applying it to network pruning task and comparing it with popular pruning methods. Table V shows the pruning results. SPATL’s parameter selection agent achieves competitive results with SoTA pruning methods, which means that it can prune redundant parameters and significantly reduce the FLOPs or pruned model with negligible accuracy loss. Moreover, state-of-the-art salient
TABLE V: Pruning comparison: top-1 accuracy results for pruned ResNets.

| Model    | Method   | FLOPs ↓ | Top-1 | ΔTop-1 |
|----------|----------|---------|-------|--------|
| Uniform  | 50%      | 84.00   |       | −7.73  |
| AMC [64] | 50%      | 86.40   |       | −5.33  |
| ResNet-20| AGMC [65]| 50%     | 88.42 | −3.31  |
|          | SFP [70] | 42%     | 90.83 | −1.37  |
|          | FPGM [71]| 42%     | 91.09 | −1.11  |
|          | DSA [72] | 50%     | 91.38 | −0.79  |
|          | SPATL (Ours) | 51%     | 91.31 | −0.42  |

parameter selection methods, such as SFP [70], DSA [72], and FPGM [71], are usually non-transferable for a given model. They require time-consuming search and re-training to find a target model architecture and salient parameters. For instance, as shown in [73] (table 2), a combined model compression method needs 85.08 lbs of CO₂ emissions to find a target model architecture. This makes it expensive to deploy on edge devices. In SPATL, the RL agent is a tiny GNN followed by an MLP. The cost to compute target salient parameters within one-shot inference (0.36 ms on NVIDIA V100) and the memory consumption is 26 KB, which is acceptable on edge devices.

2) Transfer Learning vs. No Transfer Learning: SPATL transfers the shared encoder to local non-IID datasets and addresses the heterogeneous issue of FL. To investigate the effects of transfer learning on SPATL, in this section, we disable SPATL’s transfer learning. Figure 9 (a) shows the results. We train the ResNet-20 [67] on CIFAR-10 [69] with 10 clients and sample all the clients in communication. SPATL without transfer learning has a poor performance when optimizing the model. Combining the results present in Figure 6, we can infer that a uniform model deployed on heterogeneous clients can cause performance diversity (i.e., the model performs well on some clients but poor on others). Intuitively, clients with data distribution similar to global data distribution usually perform better; nevertheless, clients far away from global data distribution are hard to converge. It is adequate to show that by introducing transfer learning, SPATL can better deal with heterogeneous issues in FL. Transfer learning enables every client to customize the model on its non-IID data and produces significantly better performance than without transfer learning.

3) Impact of Gradient Control: SPATL maintains control variates both in the local and cloud environment to help correct the local update directions and guide the encoder’s local gradient towards the global gradient direction. Figure 9 (b) shows the results of SPATL with and without gradient control. We train VGG-11 [66] on CIFAR-10 [69] with 10 clients. Training the model in the heterogeneous non-IID local dataset typically causes high variants of local gradients leading to poor convergence. The gradient control variates in SPATL maintain the global gradient direction and correct the gradient drift, thus producing a stable training process. The results are in line with our expectations that gradient control remarkably improves the training performance of SPATL.

4) Fine-tuning Reinforcement Learning Agent: This section discusses the cost of pre-training a reinforcement learning agent and the cost of customization by slightly fine-tuning the agent’s weights through online reinforcement learning. We pre-train the RL-agent and perform network pruning on ResNet-56, then transfer the agent to ResNet-18 by fine-tuning, and only update the predictor of RL-agent’s policy network. Figure 10 shows the average reward the RL agent gets on network pruning task and the agent’s corresponding update round. In both ResNet-18 and ResNet-56, the RL agent converges rapidly around 40 rounds of RL policy updating. Particularly, in ResNet-18, by slightly fine-tuning the RL-agent, it achieves comparable rewards to ResNet-56. That means the agent can be successfully transferred to a newly deployed model. This further shows the feasibility of fine-tuning the pre-trained salient parameter selection agent.

VI. CONCLUSION AND DISCUSSION

In this paper, we presented SPATL, a method for efficient federated learning using salient parameter aggregation and transfer learning. To address data heterogeneity in federated learning, we introduced a knowledge transfer local predictor that transfers the shared encoder to each client. We proposed a salient parameter selection agent to filter salient parameters of the over-parameterized model before communicating it with the server. As a result, the proposed method significantly decreases the communication overhead. We further leveraged a gradient control mechanism to stabilize the training process and make it more robust. Our experiments show that SPATL has a stable training process and achieves promising results. Moreover, SPATL significantly reduces the communication cost and accelerates the model inference time. The proposed approach may have poor performance on simple models. As Figure 4 shows, our approach works well on over-parameterized neural networks, such as ResNet [67] and VGG [66] net. However, when it turns to less-parameterized models, such as 2-layer CNNs, the salient parameter selection may degrade in performance, making the model converge slower than baselines. In practice, less-parameterized models are rarely used in real-world applications. Second, not all AI models are transferable. In our future work, we will continuously improve the universality of our method.

REFERENCES

[1] E. R. Jefferson and E. Trucco, “Chapter 20 - the challenges of assembling, maintaining and making available large data sets of clinical data for research,” in Computational Retinal
[37] P. Han, S. Wang, and K. K. Leung, “Adaptive gradient sparsification for efficient federated learning: An online learning approach,” in *Proc. of IEEE International Conference on Distributed Computing Systems (ICDCS)*, 2020.

[38] S. Yu, P. Nguyen, A. Anwar, and A. Jannesari, “Adaptive dynamic pruning for non-iiid federated learning,” *arXiv preprint arXiv:2106.06921*, 2021.

[39] Q. Li, B. He, and D. Song, “Model-Contrastive Federated Learning,” in *Proc. of Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[40] S. Yu, P. Nguyen, A. Anwar, L. Zhao, Y. Cheng, and H. Rangwala, “Fed-ta: a high-performance and communication-efficient federated learning system with asynchronous tiers,” in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2021, pp. 1–16.

[41] A. Shamsian, A. Navon, E. Fetaya, and G. Chechik, “Personalized federated learning using hypernetworks,” in *Proc. of International Conference on Machine Learning (ICML)*, 2021.

[42] W. Zhuang, Y. Wen, and S. Zhang, “Divergence-aware federated self-supervised learning,” in *Proc. of International Conference on Learning Representations (ICLR)*, 2022.

[43] Z. Zhu, J. Hong, and J. Zhou, “Data-free knowledge distillation for heterogeneous federated learning,” in *Proc. of International Conference on Machine Learning (ICML)*, 2021.

[44] Q. Liu, C. Chen, J. Qin, Q. Dou, and P.-A. Heng, “FedDG: Federated Domain Generalization on Medical Image Segmentation via Episodic Learning in Continuous Frequency Space,” in *Proc. of Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[45] P. Guo, P. Wang, J. Zhou, S. Jiang, and V. M. Patel, “Multi-Institutional Collaborations for Improving Deep Learning-Based Magnetic Resonance Image Reconstruction Using Federated Learning,” in *Proc. of Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[46] C. T. Dinh, N. H. Tran, and T. D. Nguyen, “Personalized federated learning with moreau envelopes,” *arXiv preprint arXiv:2006.08448*, 2021.

[47] Y. Huang, L. Chu, Z. Zhou, L. Wang, J. Liu, J. Pei, and Y. Zhang, “Personalized cross-silo federated learning on non-iiid data,” in *Proc. of AAAI Conference on Artificial Intelligence (AAAI)*, 2021.

[48] M. Zhang, K. Sapra, S. Fidler, S. Yeung, and J. M. Alvarez, “Personalized federated learning with first order model optimization,” in *Proc. of International Conference on Learning Representations (ICLR)*, 2021.

[49] A. Fallah, A. Mokhtari, and A. Ozdaglar, “Personalized federated learning: A meta-learning approach,” in *Proc. of Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[50] F. Hanzely, S. Hanzely, S. Horváth, and P. Richtárik, “Lower bounds and optimal algorithms for personalized federated learning,” in *Proc. of Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[51] K. Ozkara, N. Singh, D. Data, and S. Diggavi, “Quped: Quantized personalization via distillation with applications to federated learning,” in *Proc. of Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[52] X. Guo, Z. Liu, J. Li, J. Gao, B. Hou, C. Dong, and T. Baker, “Verifl: Communication-efficient and fast verifiable aggregation for federated learning,” *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 1736–1751, 2020.

[53] C. Wu, F. Wu, L. Lyu, Y. Huang, and X. Xie, “Communication-efficient federated learning via knowledge distillation,” *Nature communications*, vol. 13, no. 1, pp. 1–8, 2022.

[54] S. Caldas, S. M. K. Duddu, P. Wu, T. Li, J. Konecny, H. B. McMahan, V. Smith, and A. Talwalkar, “LEAF: A benchmark for federated settings,” in *Proc. of Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[55] J. Beutel, T. Topal, A. Mathur, X. Qiu, T. Parcollet, and N. D. Lane, “Flower: A friendly federated learning research framework,” *arXiv preprint arXiv:2007.14390*, 2020.

[56] H. Ludwig, N. Baracaldo, G. Thomas, Y. Zhou, A. Anwar, S. Rajamoni, Y. Ong, J. Radhakrishnan, A. Verma, M. Sinn et al., “Ibm federated learning: an enterprise framework white paper v0.1,” *arXiv preprint arXiv:2007.10987*, 2020.

[57] S. Gao, F. Huang, W. Cai, and H. Huang, “Network Pruning via Performance Maximization,” in *Proc. of Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[58] Y. Liu, L. Liu, C. Lin, Z. Dong, and W. Wang, “Learnable Motion Coherence for Correspondence Pruning,” in *Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2021.

[59] Z. Wang, C. Li, and X. Wang, “Convolutional Neural Network Pruning With Structural Redundancy Reduction,” in *Proc. of Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[60] B. Li, B. Wu, J. Su, and G. Wang, “EagleEye: Fast sub-net evaluation for efficient neural network pruning,” in *Proc. of the European Conference on Computer Vision (ECCV)*, 2020.

[61] T.-W. Chin, R. Ding, C. Zhang, and D. Marculescu, “Towards efficient model compression via learned global ranking,” in *Proc. of Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[62] S. Yu, A. Mazaheri, and A. Jannesari, “Topology-aware network pruning using multi-stage graph embedding and reinforcement learning,” in *Proc. of the International Conference on Machine Learning (ICML)*, ser. Proceedings of Machine Learning Research, vol. 162. PMLR, 17–23 Jul 2022, pp. 25656–25667.

[63] ——, “GNN-RL compression: Topology-aware network pruning using multi-stage graph embedding and reinforcement learning,” *arXiv preprint arXiv:2102.03214*, 2021.

[64] Y. He, J. Lin, Z. Liu, H. Wang, L.-J. Li, and S. Han, “AMC: AutoML for model compression and acceleration on mobile devices,” in *Proc. of European Conference on Computer Vision (ECCV)*, 2018.

[65] S. Yu, A. Mazaheri, and A. Jannesari, “Auto graph encoder-decoder for neural network pruning,” in *Proc. of International Conference on Computer Vision (HCC)*, 2021.

[66] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” in *Proc. of International Conference on Learning Representations (ICLR)*, 2015.

[67] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proc. of Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[68] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” *arXiv preprint arXiv:1707.06347*, 2017.

[69] A. Krizhevsky and G. Hinton, “Learning multiple layers of features from tiny images,” Master’s thesis, Department of Computer Science, University of Toronto, 2009.

[70] Y. He, G. Kang, X. Dong, Y. Fu, and Y. Yang, “Soft filter pruning for accelerating deep convolutional neural networks,” in *International Joint Conference on Artificial Intelligence (IJCAI)*, 2018, pp. 2234–2240.

[71] Y. He, P. Liu, Z. Wang, Z. Hu, and Y. Yang, “Filter pruning via geometric median for deep convolutional neural networks acceleration,” in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[72] X. Ning, T. Zhao, W. Li, P. Lei, Y. Wang, and H. Yang, “DSA: More efficient budgeted pruning via differentiable sparsity allocation,” in *Proc. of European Conference on Computer Vision (ECCV)*, 2020.

[73] T. Wang, K. Wang, H. Cai, J. Lin, Z. Liu, H. Wang, Y. Lin, and S. Han, “Amp: Joint search for network architecture, pruning and quantization policy,” in *Proc. of Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 2078–2087.
### Summary of the Experiments Reported

Our codes have been published at Artifact 1. To reproduce our experiment, by simply following the instruction shown in Artifact 1. Below are detailed experimental and hyperparameter settings.

We follow the standard Non-IID benchmark (Artifact 2) federated learning setting and implementation. We experiment on different clients and sample ratio (percentage of participating clients in each round) setting, from 10 clients to 100 clients, and the sample ratio from 0.4 to 1. During local updates, each client updates 10 rounds locally. The detailed setting can be found in supplementary materials.

We use Stochastic Gradient Descent (SGD) as the local solver with the learning rate $\alpha = 0.01$ and set the local controlling momentum parameter $\lambda$ as 0.9. Other hyperparameters during training are as follow: local epoch $E = 10$, batch size = 64.

For the salient parameter selection RL agent, we follow the experiment setting on GNN-RL (Artifact 3). The RL agent is pre-trained on ResNet-56 by a network pruning task. Fine-tuning the RL agent in the first 10 communication rounds with 20 epochs in each updating round. The discount factor is $\gamma = 0.99$, the clip parameter is 0.2, and the standard deviation of actions is 0.5. Adam optimizer is applied to update the RL agent, where the learning rate is $3 \times 10^{-4}$ and the $\beta = (0.9, 0.999)$.

1. Algorithm training and execute platform: We test our codes on NVIDIA RTX 6000s and Tesla V100s separately and independently.
2. Operating systems and versions: RedHat Enterprise Linux 7
3. Compilers and version: Python >=3.8
4. Algorithm required dependencies and packages: PyTorch >=1.8.0 (CUDA 11.1), torchvision 0.7.0, torch-geometric >= 1.6.1, scikit-learn 0.22.1, numpy 1.18.1, scipy 1.4.1, pandas 0.24.2, requests 2.23.0

### Author-Created or Modified Artifacts:

**Artifact 1**
- Persistent ID: https://zenodo.org/badge/latestdoi/429851236
- Artifact name: SPATL

**Artifact 2**
- Persistent ID: https://github.com/Xtra-Computing/NIID-Bench
- Artifact name: NIID-Bench

**Artifact 3**
- Persistent ID: https://gnn-rl.readthedocs.io/en/latest/Installation
- Artifact name: GNN-RL

**Reproduction of the artifact with container:** We provide Docker container images for the reproduction of experiments. Detailed instruction can be found in Artifact 1, under the Docker directory, README.md. We also provide instruction on running experiments in conda virtual environment.