Towards Collaborative Intelligence: Routability Estimation based on Decentralized Private Data

ABSTRACT

Applying machine learning (ML) in design flow is a popular trend in EDA with various applications from design quality predictions to optimizations. Despite its promise, which has been demonstrated in both academic researches and industrial tools, its effectiveness largely hinges on the availability of a large amount of high-quality training data. In reality, EDA developers have very limited access to the latest design data, which is owned by design companies and mostly confidential. Although one can commission ML model training to a design company, the data of a single company might be still inadequate or biased, especially for small companies. Such data availability problem is becoming the limiting constraint on future growth of ML for chip design. In this work, we propose an Federated-Learning based approach for well-studied ML applications in EDA. Our approach allows an ML model to be collaboratively trained with data from multiple clients but without explicit access to the data for respecting their data privacy. To further strengthen the results, we co-design a customized ML model FLNet and its personalization under the decentralized training scenario. Experiments on a comprehensive dataset show that collaborative training improves accuracy by 11% compared with individual local models, and our customized model FLNet significantly outperforms the best of previous routability estimators in this collaborative training flow.

1 INTRODUCTION

Electronic design automation (EDA) techniques have achieved remarkable progress over past decades. However, the current chip design flow is still largely restricted to individual point tools with limited interplay across different tools and design steps. Tools in early steps cannot well judge if their solutions may eventually lead to satisfactory designs, and the consequence of a poor solution cannot be found until very late. Such disjointedness in the design flow is traditionally mitigated by either simplified estimations with heuristics or iterative design, which often lead to over-conservative design or longer turn-around time, respectively. To improve the predictability in chip design flow, machine learning (ML) models have been constructed based on prior data to provide early feedback or help accelerate the solving of EDA problems.

In recent years, ML for EDA has become a trending topic [13]. ML models are applied at almost all design stages of the VLSI design flow, including high-level synthesis, logic synthesis, and physical design [2, 8, 24, 28], making predictions on timing [2], power [25], routability [8, 24, 28], etc. These ML models learn from prior solutions and typically provide orders-of-magnitude faster design quality evaluations. Besides being a hot research topic in academia, ML-based estimators have also gained popularity in the EDA industry. Recent versions of commercial tools already support the construction of ML models on delay [6] or congestion predictions [23]. The vendors also claim improved PPA or faster convergence after invoking the ML models in their tools [6, 23]. In summary, ML for EDA has demonstrated its impressive contribution to the quality of result (QoR) and overall turn-around time in both academia and industry.

However, despite the proven advantages, there are still obstacles that prevent wide applications of ML models in EDA. One vital but rarely explicitly discussed challenge is the availability of data. The data includes circuit designs and corresponding chip qualities including power, performance, and etc. Most latest design data are owned by design companies and are highly confidential. As a result, many works from academia have to construct models with designs from public benchmarks, most of which are either outdated or over-simplified only for contest purposes. Data availability also affects the quality of practical ML for EDA models in industry. Although one can train ML models within a design company, the data of a single company might still be inadequate or biased, especially for small companies. As for EDA vendors, due to data privacy concerns, some commercial tools [6] have to construct ML models from scratch, only using the limited raw data provided by each client. In summary, lack of data has seriously hampered the adoption of ML models in EDA, yet very limited research explorations can be found for mitigating this problem.

To address the above challenges and promote collaborative intelligence, we present a novel framework to collaboratively train ML models without explicitly collecting or viewing data from decentralized design owners. This framework is demonstrated on routability estimation, which is a well-studied problem in ML for EDA. To promote collaborative learning, we co-design the federated learning ML model architecture, named FLNet, and its personalization techniques to seek unprecedented solutions in federated learning. The proposed approach, when put in practice, allows design companies to leave private data on their own servers and admit only pre-determined operations and communications[1]. In this way, design companies can jointly utilize their design data for ML model construction without disclosure of their product. For example, EDA vendors can first train a significantly more general ML model using designs from all its cooperators. Then each cooperator, as a client, can optionally customize the general model for itself by leveraging personalization techniques (e.g. local fine-tuning) using its own private data to achieve better performance.

Our contributions in this work are summarized as follows.

- We bring attention to the data availability problem in ML for EDA, and propose a collaborative training solution to encourage data sharing while respecting data privacy.
- We co-design the personalization flow and network architecture for our proposed decentralized training scenario. With the best personalization, FLNet outperforms previous routability estimators by 11% in accuracy[2] on a comprehensive dataset with 7,131 layouts from 74 different designs.
- We provide detailed ablation studies to justify our co-design on personalization and network architectures.

[1] Engineering details including the implementation of FL framework [3, 12] and privacy concerns [20, 22] about the framework on both model and data have been well studied for general machine learning tasks. These engineering details are not special in ML for EDA, thus are not the focus of this paper.

[2] In order to contribute to the reproducibility and fair comparisons in ML for EDA, this framework and the related dataset are anonymously open-sourced in https://anonymous.4open.science/r/Decentralized-Routability-Estimation-F509/.

2 PRELIMINARIES

2.1 Routability Estimation

We demonstrate our algorithm on routability estimation, since it is a representative and well-studied topic [8, 17, 24, 28] in ML for EDA, and its problem formulation and solutions share many similarities with other important EDA problems like IR drop estimation [10], clock tree prediction [18], lithography hotspot detection [14, 27], optical proximity correction (OPC) [26], etc. Previous works on these problems typically borrow ideas from computer vision and adopt deep learning techniques including convolutional neural network (CNN) or generative adversarial network (GAN) to process the features from circuit layouts.

Previous routability estimators use either routing congestions [8, 28] or DRC (design rule checking) [17, 24] as the metric of routability. They detect congestion locations or DRC hotspots. Given a set of placement solutions with extracted input feature maps X_i, routability estimators generate a neural network model f to detect the locations of DRC hotspots or congestions Y_i:

$$f : X_i \in \mathbb{R}^{w \times h \times c} \rightarrow Y_i \in \{0, 1\}^{w \times h}$$

where w and h are the width and height of the layout, and c indicates the number of input features/channels.

We demonstrate our learning algorithm through comparisons with two representative routability estimators from RouteNet [24] and PROS [8]. Both works adopt fully convolutional network (FCN)-based estimators with significantly different model structures. The estimator from RouteNet [24], as an earlier work, consists of only convolution layers, trans-convolutional layers and a shortcut structure. In comparison, the estimator from PROS [8] adopts more advanced structures including dilated convolution [29] blocks, refinement blocks, and sub-pixel upsampling blocks.

2.2 Federated Learning

Federated learning (FL) includes a series of decentralized training techniques, proposed for their distinct privacy protection advantage compared with training on a central machine with persistent data [19]. FedAvg [19] is a popular FL algorithm for most computer vision tasks. In FedAvg, the decentralized training is performed iteratively. In each round, the clients send updates of locally trained models to the central server, and the server then averages the collected updates and distributes the aggregated update back to all the clients. FedAvg works well with independent and identically distributed (IID) datasets but may suffer from significant performance degradation when it is applied to non-IID datasets.

3 PROBLEM FORMULATION

We assume there are altogether K clients providing their data for model training. In practice, the circuit designs from the same client/company tend to be more similar to each other, since they may be from the same series of products, while different clients may provide largely different designs. This is reflected in our experiment by only assigning designs from the same benchmark to the same client. Assume each client $k \in [1, K]$ provides n_k data samples. For the routability estimation task, each data sample includes one placement solution, and its label is the ground-truth DRC hotspot map. The training data for each client k is denoted as $\{X_i, Y_i\}_k$ $(i \in [1, n_k], k \in [1, K])$, where $X_i \in \mathbb{R}^{w \times h \times c}$ is the feature map of a placement with $w \times h$ grids and c channels, while $Y_i \in \{0, 1\}^{w \times h}$ is the hotspot distribution. To verify the estimator accuracy, each client also has their own testing data $\{X_i^{Test}, Y_i^{Test}\}_k$ $(k \in [1, K])$, which is generated from circuit designs completely different from those of the training data. Traditionally, if the developer directly commission ML model training to each client, it results in K local models trained only by each client's local data. These models are baselines denoted as b_k, which is trained on $\{X_i, Y_i\}_k$ by the client k.

A key difference in our problem setting compared with previous works is the *data privacy* constraint. All clients' data should be private to themselves, which means no party other than the client itself should have the access to its training and testing data. Under the data privacy constraint, there are two goals in this work. First, we develop a general model that achieves higher performance on all K clients' testing data $\{X_i^{\text{Test}}, Y_i^{\text{Test}}\}_k (k \in [1, K])$. This model should generalize better than local model baselines. Second, we build a customized model for each client for better local accuracy.

4 METHODOLOGY

In this work, we try to train our model for routability estimation utilizing data from all K clients without violating their data privacy. Compared with baselines trained on each local client, we enlarge the training dataset by K times. To achieve this, we need a strict decentralized training setting with limited data access for privacy protection. And the decentralized training setting poses challenges to successful model construction. The client-level data heterogeneity commonly seen in routability estimation tasks makes decentralized training suffers from convergence issues and performance degradation. To make thing worse, existing models are too complex and involve special operators that are not robust against some operations in the decentralized training setting, thus failing to perform well. Therefore, in the following sections, we 1) analyze the current challenges and a solution (FedProx) of decentralized training for routability prediction; 2) propose **FLNet**, a novel model customized for better performance in the limited decentralized training setting; 3) explore **model personalization techniques** to alleviate the negative impact of client-level data heterogeneity; 4) briefly mention the features we use for routability estimation.

4.1 When Decentralized Training meets Routability Estimation

The decentralized training setting requires all training and optimization operations to take place only at the client side, without gathering data from any client. And the developer can only receive model parameters from its clients, perform parameter aggregation, and deploy the average parameters back to the clients. Figure 1 shows the visualization of the decentralized training setting with K clients. For each round $r \in [1, R]$, each client k trains the model on its own data, and send the trained parameters w_k^r to the developer. Then the developer performs parameter aggregation on all collected parameters and generate an average model $W^{r+1} = \sum_{k=1}^{K} \frac{n_k}{n} w_k^r$. To conclude round r, the average model W^{r+1} is deployed back to all the clients for further training at the next round. This procedure is repeated for R times and the developer will construct a generalized model W^R, which is the aggregated model at the R-th round.

This setting brings at least two new challenges. First, **high data heterogeneity among clients hinders the convergence of decentralized training**. Different clients can contain largely different circuit designs in terms of functionality or microarchitecture. The

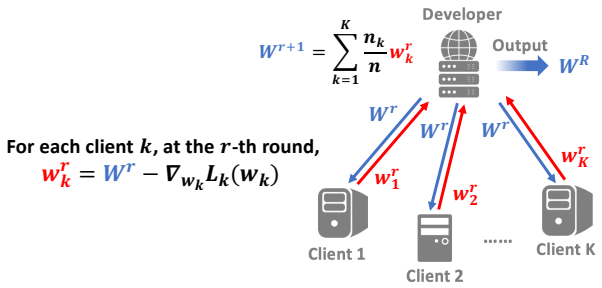


Figure 1: The visualization of the decentralized training setting. R denotes the total number of rounds. K denotes the number of clients. w and W denotes the locally trained models and the aggregated models, respectively.

intrinsic differences between circuit designs cause the data heterogeneity in their feature distributions. And ML models trained on different clients tend to capture non-general patterns which reflect such data heterogeneity. The typical client-level heterogeneity of routability data makes decentralized training suffer from slow convergence and low accuracy, or even fail to converge.

Second, **existing routability estimator models do not cooperate well with decentralized training**, leading to a large performance degradation. Existing works typically utilize complex models with very high non-linearity, and thus are over-sensitive to operations like parameter aggregation, which happens frequently in the decentralized training setting. Besides, they also involve model components that increase convergence difficulty in the decentralized training setting.

We apply the FedProx [15] method to address the convergence issue arising from heterogeneous training data distribution among clients. In FedProx, each client performs training on its local data, and sends the trained model parameters instead of the data to the central machine of the model developer. The model developer then performs parameter aggregation based on the collected parameters from all clients, and sends back the average parameters to the clients for further training. Such procedure is repeated until the model converges. FedProx's training setting satisfies the requirements of the aforementioned decentralized training scenarios. Furthermore, to alleviate the data heterogeneity challenge, it adds an extra proximal term to penalize the difference between each local model and the model received from the developer at the beginning of each training round. This makes FedProx applicable to EDA tasks such as routability estimation, where heterogeneous data is very common due to huge difference in circuits designs. In FedProx, at the r-th round, client k optimizes the following objective L_{Prox}:

$$\min_{w_k} L_{\text{Prox}}(w_k, W^r) = \sum_{i=1}^{n_k} (w_k(X_i) - Y_i)^2 + \mu ||W^r - w_k||^2, \quad (1)$$

where μ is a hyper-parameter that controls the contribution of the proximal term. In this way, the difference between the global model W^r and local model w_k is constrained during the local training, thus preventing the divergence of local models.

4.2 FLNet: Routability Model Customized for FL

Existing works typically fail to adapt to the decentralized training setting because they utilize complex models that have much higher non-linearity than simple models. Such high non-linearity introduces low robustness against fluctuation of model parameters. And in the decentralized training setting, the model parameter fluctuation frequently happens at the parameter aggregation operation. This can lead to much lower performance of the model compared with the same model trained in a centralized setting. On the other hand, existing models commonly come with a large number of sequential layers, and thus need special operators like Batch Normalization layers. Batch Normalization layers aim to ensure good convergence by whitening the input of each layer using recorded mean and variance during training. However, when training a model with Batch Normalization layers using decentralized training setting, Batch Normalization layers usually fail to obtain stable records of mean and variance due to the frequent model parameter aggregation operation, which makes the model even harder to converge. As a result, existing models for routability estimation generally fail to adapt well to the limited decentralized training setting and show large performance degradation.

Table 1: FLNet Model Architecture Configuration

Layer	Kernel size	#Filters	Activation
input_conv	9×9	64	ReLU
output_conv	9×9	1	None

To achieve the best performance in federated learning, we co-designed FLNet to reduce the performance degradation commonly seen in existing models for routablity estimation. FLNet is a 2-layer CNN model without any Batch Normalization operators. It has much fewer model parameters than previous works and thus is more robust to the negative impact from parameter fluctuation introduced by parameter aggregation. Particularly, this parameter fluctuation is amplified by the client-level data heterogeneity commonly seen in routability data. FLNet's robustness can protect its performance from unacceptable degradation, making it outperform existing complex models when the data is highly heterogeneous. Table 1 shows detailed configuration of FLNet. Despite its simple structure, we select a large kernel size 9×9 for both layers to ensure a large receptive field at the output. Therefore, FLNet can still capture features with a relatively large spatial range, which is important in routability estimation.

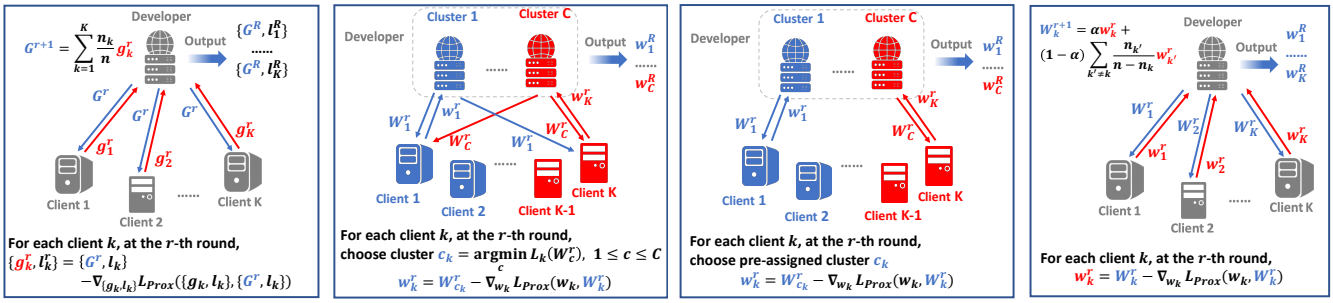
4.3 Personalization in Federated Learning

Federated learning methods (e.g., FedProx [15]) train one generalized model to achieve good average accuracy on all clients. But personalization techniques in federated learning aim to train models that can perform better than the generalized model for individual clients that are highly data and system heterogeneous. Typically, design companies care more about a model's accuracy on the local data than its transferability or generality. Actually, we can further utilize model personalization techniques to trade off extra training cost and model generality for improvement of local accuracy.

In this section, we explore a set of federated learning personalization techniques based on the FedProx training scheme. We first explore pre-existing techniques (e.g., FedProx-LG [16] and Iterative Federated Clustering Algorithm (IFCA) [11]) as follows:

FedProx-LG is based on the prior idea of [16]. Figure 2(a) shows the visualization of FedProx-LG. FedProx-LG partitions a model into a global part g and a local part l. At the r-th round, the model developer only communicate and aggregates the global part g^r, leaving the local part l_k^r, $(1 \le k \le K)$ private to each client k.

Iterative Federated Clustering Algorithm (IFCA) introduces client-level clustering to alleviate the negative effect of data heterogeneity. Figure 2(b) shows the visualization of IFCA. The K clients can be categorized as C clusters, and the clients of the same cluster have higher similarity and more shareable features to be captured by the DNN model. In IFCA, the model developer initializes a model



(a) (b) (c) (d)

Figure 2: Visualizations of federated learning personalization techniques based on FedProx. g and l denote locally trained global parts and local parts, respectively. G denotes aggregated global parts. C denotes the number of clusters. (a) FedProx-LG. (b) Iterative federated clustering algorithm (IFCA). (c) Assigned clustering. (d) FedProx + α-portion sync.

for each cluster respectively. At each round r, client k determines its cluster c_k by verifying the loss $L_k(W_c^r)$ of C cluster models ($c = 1, 2, \ldots, C$) on client k's training data and chooses the cluster c with the lowest loss. Then, client k trains the chosen cluster model $W_{c_k}^r$ and sends the updated model w_k^r to the developer. At the developer side, each cluster only receives and aggregates the models from its corresponding clients at that round. The procedure repeats and the model clustering is updated iteratively.

Besides existing personalization techniques, we make our exploration comprehensive by investigating additional techniques such as local fine-tuning, assigned clustering, and α-portion sync to further enhance FL personalization, demonstrated as follows:

Local fine-tuning is a simple but effective technique for model personalization. Based on the model trained with FedProx, each client can further fine-tune the model received from the developer for extra steps on its own data. In this way, all clients make the collaboratively-trained model adapt towards the distribution of their respective local data.

Assigned clustering pre-assigns a cluster for each client by leveraging prior knowledge about the clients' similarity. Figure 2(c) shows the visualization of the assigned clustering method. Compared with IFCA, this method assigns a fixed cluster c_k to each client k for all rounds, and directly gains faster convergence and potentially higher accuracy.

α-portion sync is another personalization technique with much less extra cost, where the model developer simply performs different weighted aggregation for model parameters from each client. Figure 2(d) shows the visualization of the α-portion sync algorithm. At the beginning of each round r, for each client k, the developer aggregates a customized model W_k^r that takes client k's previous parameters w_k^{r-1} as α-portion in the aggregation. Compared with FedProx, the α-portion sync method puts higher weight to each client's own parameters in the parameter aggregation. And thus, it customizes the model to adapt more to the local data, while also learning from data of other clients.

4.4 Feature Extraction

Regarding the features for routability estimation, we follow previous works [7, 8, 24] to select features and perform feature extraction. More specifically, our selected features capture both cell density features (e.g., locations of cells) and wire density features (e.g., connectivity between instances). Cell density features include the routing blockage information. Wire density features encode the instance connectivity and routing congestion information using several heuristics, such as RUDY [24] and fly lines.

5 EXPERIMENT RESULTS

5.1 Experiment Setup

We construct a comprehensive dataset using 74 designs with largely varying sizes from multiple benchmarks. There are 29 designs from ISCAS'89 [4], 13 designs from ITC'99 [9], 19 other designs from Faraday and OpenCores in the IWLS'05 [1], 13 designs from ISPD'15 [5]. For each design, multiple placement solutions are generated with different logic synthesis and physical design settings. Altogether 7,131 placement solutions are generated from these 74 designs. We apply Design Compiler® for logic synthesis and Innovus® [6] for physical design, with the NanGate 45nm technology library [21].

To validate our algorithm, we mimic a real application scenario by splitting all designs to nine different clients ($K = 9$). Since designs from the same client tend to be more similar to each other, we assign designs from the same benchmark suite to the same client. Then for each client, we randomly split around 70% of designs to be training data and the other 30% of designs to be the testing data. Notice that there are no clients sharing common designs, and there are no designs belonging to training and testing data at the same time. This prevents information leakage between clients and ensures testing designs are completely unseen to trained models. Details of the design assignment and number of placements in each client are shown in Table 2. Three clients collect designs from ITC'99, three clients collect designs from ISCAS'89, two clients collect from Faraday and OpenCores in the IWLS'05, and one client collects from ISPD'15. For each client, the number of placement solutions in the training data ranges from 175 to 812, and the number in the testing data ranges from 84 to 348.

Table 2: Experiment Data Setup for Each Client

Clients	Training Designs (Num of Placements)	Testing Designs (Num of Placements)
Client 1	4 designs in ITC'99 (462)	2 designs in ITC'99 (230)
Client 2	2 designs in ITC'99 (231)	1 design in ITC'99 (114)
Client 3	2 designs in ITC'99 (231)	2 designs in ITC'99 (232)
Client 4	7 designs in ISCAS'89 (812)	3 designs in ISCAS'89 (348)
Client 5	7 designs in ISCAS'89 (812)	3 designs in ISCAS'89 (348)
Client 6	6 designs in ISCAS'89 (697)	3 designs in ISCAS'89 (348)
Client 7	6 designs in IWLS'05 (656)	3 designs in IWLS'05 (280)
Client 8	7 designs in IWLS'05 (742)	3 designs in IWLS'05 (329)
Client 9	9 designs in ISPD'15 (175)	4 designs in ISPD'15 (84)

Table 3: Testing Accuracy Comparison (ROC AUC) on Routability Prediction with FLNet

	Testing on									
	Client 1	Client 2	Client 3	Client 4	Client 5	Client 6	Client 7	Client 8	Client 9	**Average**
Local Average (b_1 to b_9)	0.76	0.75	0.71	0.72	0.67	0.70	0.76	0.64	0.82	**0.72**
Training Centrally on All Data	0.87	0.87	0.77	0.80	0.75	0.77	0.82	0.70	0.92	**0.81**
FedProx	0.82	0.78	0.73	0.75	0.72	0.74	0.82	0.69	0.96	**0.78**
FedProx-LG	0.77	0.61	0.65	0.65	0.60	0.69	0.77	0.63	0.93	**0.70**
IFCA	0.83	0.79	0.73	0.76	0.71	0.75	0.82	0.69	0.87	**0.77**
FedProx + Fine-tuning	0.84	0.89	0.79	0.78	0.72	0.75	0.82	0.72	0.90	**0.80**
Assigned Clustering	0.81	0.86	0.75	0.76	0.72	0.75	0.81	0.70	0.88	**0.78**
FedProx + α-Portion Sync	0.82	0.79	0.73	0.76	0.72	0.75	0.81	0.69	0.90	**0.78**

Table 4: Testing Accuracy Comparison (ROC AUC) on Routability Prediction with RouteNet [24]

	Testing on									
	Client 1	Client 2	Client 3	Client 4	Client 5	Client 6	Client 7	Client 8	Client 9	**Average**
Local Average (b_1 to b_9)	0.76	0.76	0.71	0.73	0.68	0.71	0.75	0.64	0.78	**0.73**
Training Centrally on All Data	0.86	0.88	0.79	0.82	0.81	0.77	0.82	0.75	0.94	**0.83**
FedProx	0.63	0.83	0.71	0.72	0.66	0.67	0.63	0.57	0.42	**0.65**
FedProx-LG	0.60	0.55	0.57	0.50	0.51	0.49	0.54	0.52	0.46	**0.53**
IFCA	0.46	0.28	0.35	0.37	0.39	0.44	0.43	0.43	0.71	**0.43**
FedProx + Fine-tuning	0.83	0.86	0.76	0.75	0.74	0.75	0.81	0.72	0.90	**0.79**
Assigned Clustering	0.70	0.85	0.74	0.65	0.64	0.65	0.49	0.46	0.89	**0.67**
FedProx + α-Portion Sync	0.66	0.57	0.61	0.57	0.54	0.58	0.68	0.58	0.72	**0.61**

Table 5: Testing Accuracy Comparison (ROC AUC) on Routability Prediction with PROS [8]

	Testing on									
	Client 1	Client 2	Client 3	Client 4	Client 5	Client 6	Client 7	Client 8	Client 9	**Average**
Local Average (b_1 to b_9)	0.65	0.63	0.61	0.61	0.58	0.62	0.66	0.59	0.72	**0.63**
Training Centrally on All Data	0.75	0.68	0.65	0.65	0.62	0.62	0.73	0.65	0.73	**0.67**
FedProx	0.67	0.60	0.61	0.64	0.63	0.64	0.65	0.59	0.58	**0.62**
FedProx-LG	0.69	0.62	0.62	0.63	0.61	0.65	0.71	0.60	0.84	**0.66**
IFCA	0.50	0.58	0.52	0.53	0.51	0.48	0.51	0.51	0.35	**0.50**
FedProx + Fine-tuning	0.74	0.65	0.76	0.72	0.53	0.67	0.81	0.69	0.50	**0.67**
Assigned Clustering	0.47	0.55	0.51	0.48	0.49	0.51	0.70	0.60	0.36	**0.52**
FedProx + α-Portion Sync	0.64	0.45	0.56	0.58	0.55	0.52	0.64	0.55	0.59	**0.56**

We evaluate accuracy with receiver operating characteristic (ROC) area under curve (AUC), measured based on the confusion matrix from prediction. The AUC ranges from 0 to 1, with larger value indicating higher model accuracy. We verify the accuracy in ROC AUC for all proposed federated learning methods with personalization using three models, two representitive routability estimators RouteNet [24] and PROS [8], and our proposed FLNet.

For hyperparameters in our experiment, the number of rounds $R = 50$, the number of model update steps in each round $S = 100$, the number of steps for local fine-tuning $S' = 5000$. We use a learning rate of 0.0002, Adam optimizer, and an L2 regularization strength of 0.00001. The FedProx proximal term strength μ is 0.0001. For the α-portion model, we test $\alpha = 0.5$. For FedProx-LG, we set the output layers of the three models to be the local part, while the remaining layers to be the global part. For IFCA, we set the number of clusters $C = 4$. For the assigned clustering method, we select 4 clusters: Client 1-3, Client 4-6, Client 7-8, and Client 9.

5.2 Training Method Evaluation

Table 3 shows the accuracy of FLNet, using various model training algorithms based on decentralized private data. The first baseline is the average performance among $K = 9$ locally-trained models b_1 to b_9, which equals 0.72. This corresponds to the performance of traditional ML model construction methods in such a decentralized setting. In addition, a very strong baseline is training the ML model centrally on all training data. This indicates a scenario where we can explicitly collect all data together from all clients without data privacy concerns. Without accuracy degradation caused by heterogeneity among different clients, this accuracy 0.81 can be viewed as an empirical upper limit we should target for decentralized training. Ideally, our FL algorithms should achieve the same accuracy if they well address all challenges in the decentralized setting.

As Table 3 shows, our FLNet model trained with FedProx reaches the accuracy of 0.78, outperforming the locally trained baselines (b_1 to b_9) by 0.06 in absolute accuracy value. When inspecting its

performance on each individual client, it outperforms the baseline's performance on every single client. This FedProx based on FLNet is our proposed method to generate a single generalized model.

To achieve better local accuracy, instead of simply adopting the generalized model from FedProx, we apply different personalization techniques to customize the model for each client. Among all five different personalization algorithms we proposed, the straightforward FedProx + Fine-tuning achieves the best accuracy in 0.80, outperforming local models by 0.08, which is 11% relative improvement. This accuracy is also very close to the empirical upper limit accuracy (0.81) from centralized training. Compared with simple FedProx, it further improves accuracy at the cost of extra fine-tuning for each client.

5.3 ML Model Evaluation

The accuracy of representative routability models RouteNet [24] and PROS [8] are shown in Table 4 and Table 5, respectively. Compared with FLNet, RouteNet in Table 4 achieves slightly better accuracy for local training and centrally training. This is reasonable since the RouteNet model is developed under this traditional training setup. In comparison, for all decentralized training algorithms we proposed, FLNet achieves superior accuracy than RouteNet. This proves the robustness of our proposed FLNet against client data heterogeneity, thus FLNet is a better choice for federated learning.

As indicated by Table 4, the FedProx method based on RouteNet is obviously vulnerable to decentralized training and is even less accurate than local models (b_1 to b_9). Similarly, most personalization methods are also significantly affected. Only after finetuning on local clients, those customized models' accuracy rises back to 0.79. This is because such finetuning process is no longer under the decentralized setting.

In Table 5, the overall accuracy of PROS, another popular routability model, is much lower than RouteNet and FLNet. One possible reason is PROS is originally proposed to predict routing congestions, and thus is less effective for DRC violation prediction. Also, its higher model complexity makes it even more vulnerable to the data heterogeneity. But we can still observe a trend very similar to RouteNet. The FedProx model based on PROS is also less accurate than local models (b_1 to b_9), and after finetuning on local clients, the accuracy rises close to the accuracy of centralized training.

In summary, the combination of our proposed FLNet and FedProx can well utilize decentralized private data and achieve better performance (0.78) than existing methods RouteNet and PROS with the accuracies 0.73 and 0.63, respectively. By further fine-tuning the model on each client, the averaged accuracy of FLNet with FedProx + Fine-tuning rises to 0.80, which is close to the accuracy of training all data centrally.

6 CONCLUSION

In this work, we bring attention to the serious data availability problem in ML for EDA applications, and propose a federated learning-based solution to encourage data sharing. Our proposed collaborative training method based on FLNet model proves to be robust to the client heterogeneity in decentralized training data. This solution demonstrates its benefit in routability prediction and can be potentially extended to other ML for EDA tasks, especially layout-level predictions and optimizations with similar problem formulations. In the future, as the ML methods get more widely deployed in design flow and the demand on training data keeps increasing, such collaborative training may become a standard training operation.

REFERENCES

[1] Christoph Albrecht. 2005. IWLS 2005 benchmarks. In *IWLS: http://www.iwls.org*.
[2] Erick Carvajal Barboza, Nishchal Shukla, Yiran Chen, and Jiang Hu. 2019. Machine learning-based pre-routing timing prediction with reduced pessimism. In *DAC*.
[3] Daniel J Beutel, Taner Topal, Akhil Mathur, Xinchi Qiu, Titouan Parcollet, Pedro PB de Gusmão, and Nicholas D Lane. 2020. Flower: A friendly federated learning research framework. *arXiv preprint arXiv:2007.14390* (2020).
[4] Franc Brglez, David Bryan, and Krzysztof Kozminski. 1989. Combinational profiles of sequential benchmark circuits. In *ISCAS*.
[5] Ismail S Bustany, David Chinnery, Joseph R Shinnerl, and Vladimir Yutsis. 2015. ISPD 2015 benchmarks with fence regions and routing blockages for detailed-routing-driven placement. In *ISPD*.
[6] Cadence. 2021. Innovus Implementation System. https://www.cadence.com/en_US/home/tools/digital-design-and-signoff/soc-implementation-and-floorplanning/innovus-implementation-system.html
[7] Chen-Chia Chang, Jingyu Pan, Tunhou Zhang, Zhiyao Xie, Jiang Hu, Weiyi Qi, Chunwei Lin, Rongjian Liang, Joydeep Mitra, Elias Fallon, and Yiran Chen. 2021. Automatic Routability Predictor Development Using Neural Architecture Search. In *ICCAD*.
[8] Jingsong Chen, Jian Kuang, Guowei Zhao, Dennis J-H Huang, and Evangeline FY Young. 2020. PROS: A Plug-in for Routability Optimization applied in the State-of-the-art commercial EDA tool using deep learning. In *ICCAD*.
[9] Fulvio Corno, Matteo Sonza Reorda, and Giovanni Squillero. 2000. RT-level ITC'99 benchmarks and first ATPG results. *Design & Test of computers* (2000).
[10] Yen-Chun Fang, Heng-Yi Lin, Min-Yan Sui, Chien-Mo Li, and Eric Jia-Wei Fang. 2018. Machine-learning-based dynamic IR drop prediction for ECO. In *ICCAD*.
[11] Avishek Ghosh, Jichan Chung, Dong Yin, and Kannan Ramchandran. 2020. An efficient framework for clustered federated learning. *arXiv preprint arXiv:2006.04088* (2020).
[12] Yaochen Hu, Di Niu, Jianming Yang, and Shengping Zhou. 2019. Fdml: A collaborative machine learning framework for distributed features. In *KDD*.
[13] Guyue Huang, Jingbo Hu, Yifan He, Jialong Liu, Mingyuan Ma, Zhaoyang Shen, Juejian Wu, Yuanfan Xu, Hengrui Zhang, Kai Zhong, et al. 2021. Machine learning for electronic design automation: A survey. *arXiv preprint arXiv:2102.03357* (2021).
[14] Yiyang Jiang, Fan Yang, Bei Yu, Dian Zhou, and Xuan Zeng. 2020. Efficient Layout Hotspot Detection via Binarized Residual Neural Network Ensemble. *TCAD* (2020).
[15] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. 2018. Federated optimization in heterogeneous networks. *arXiv preprint arXiv:1812.06127* (2018).
[16] Paul Pu Liang, Terrance Liu, Liu Ziyin, Nicholas B Allen, Randy P Auerbach, David Brent, Ruslan Salakhutdinov, and Louis-Philippe Morency. 2020. Think locally, act globally: Federated learning with local and global representations. *arXiv preprint arXiv:2001.01523* (2020).
[17] Rongjian Liang, Hua Xiang, Diwesh Pandey, Lakshmi Reddy, Shyam Ramji, Gi-Joon Nam, and Jiang Hu. 2020. DRC Hotspot Prediction at Sub-10nm Process Nodes Using Customized Convolutional Network. In *ISPD*.
[18] Yi-Chen Lu, Jeehyun Lee, Anthony Agnesina, Kambiz Samadi, and Sung Kyu Lim. 2019. GAN-CTS: A generative adversarial framework for clock tree prediction and optimization. In *ICCAD*.
[19] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*. PMLR.
[20] Mohammad Naseri, Jamie Hayes, and Emiliano De Cristofaro. 2020. Toward robustness and privacy in federated learning: Experimenting with local and central differential privacy. *arXiv preprint arXiv:2009.03561* (2020).
[21] Si2. 2018. NanGate 45nm Open Cell Library. https://si2.org/open-cell-library/
[22] Jingwei Sun, Ang Li, Binghui Wang, Huanrui Yang, Hai Li, and Yiran Chen. 2020. Provable Defense against Privacy Leakage in Federated Learning from Representation Perspective. *arXiv preprint arXiv:2012.06043* (2020).
[23] Synopsys. 2021. IC Compiler II for Physical Implementation. https://www.synopsys.com/implementation-and-signoff/physical-implementation/ic-compiler.html
[24] Zhiyao Xie, Yu-Hung Huang, Guan-Qi Fang, Haoxing Ren, Shao-Yun Fang, Yiran Chen, and Jiang Hu. 2018. RouteNet: Routability prediction for mixed-size designs using convolutional neural network. In *ICCAD*.
[25] Zhiyao Xie, Xiaoqing Xu, Matt Walker, Joshua Knebel, Kumaraguru Palaniswamy, Nicolas Hebert, Jiang Hu, Huanrui Yang, Yiran Chen, and Shidhartha Das. 2021. APOLLO: An Automated Power Modeling Framework for Runtime Power Introspection in High-Volume Commercial Microprocessors. In *MICRO*.
[26] Haoyu Yang, Shuhe Li, Zihao Deng, Yuzhe Ma, Bei Yu, and Evangeline FY Young. 2019. GAN-OPC: Mask optimization with lithography-guided generative adversarial nets. *TCAD* (2019).
[27] Haoyu Yang, Luyang Luo, Jing Su, Chenxi Lin, and Bei Yu. 2017. Imbalance aware lithography hotspot detection: a deep learning approach. *Journal of Micro/Nanolithography, MEMS, and MOEMS* (2017).
[28] Cunxi Yu and Zhiru Zhang. 2019. Painting on placement: Forecasting routing congestion using conditional generative adversarial nets. In *DAC*.
[29] Fisher Yu and Vladlen Koltun. 2015. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122* (2015).