Lithography Hotspot Detection via Heterogeneous Federated Learning with Local Adaptation

Xuezhong Lin[1*], Jingyu Pan[2*], Jinming Xu[1], Yiran Chen[2] and Cheng Zhuo[1]

[1]Zhejiang University, China; [2]Duke University, USA

***Abstract*—As technology scaling is approaching the physical limit, lithography hotspot detection has become an essential task in design for manufacturability. While the deployment of pattern matching or machine learning in hotspot detection can help save significant simulation time, such methods typically demand for non-trivial quality data to build the model, which most design houses are short of. Moreover, the design houses are also unwilling to directly share such data with the other houses to build a unified model, which can be ineffective for the design house with unique design patterns due to data insufficiency. On the other hand, with data homogeneity in each design house, the locally trained models can be easily over-fitted, losing generalization ability and robustness. In this paper, we propose a heterogeneous federated learning framework for lithography hotspot detection that can address the aforementioned issues. On one hand, the framework can build a more robust centralized global sub-model through heterogeneous knowledge sharing while keeping local data private. On the other hand, the global sub-model can be combined with a local sub-model to better adapt to local data heterogeneity. The experimental results show that the proposed framework can overcome the challenge of non-independent and identically distributed (non-IID) data and heterogeneous communication to achieve very high performance in comparison to other state-of-the-art methods while guaranteeing a good convergence rate in various scenarios.**

I. Introduction

As technology scaling is approaching the physical limit, the lithography process is considered as a critical step to continue the Moore's law [1]. Even though the light wavelength for the process is larger than the actual transistor feature size, recent advances in lithography processing, $e.g.$, multi-patterning, optical proximity correction, $etc.$, have made it possible to overcome the sub-wavelength lithography gap [2]. On the other hand, due to the complex design rules and process control at sub-14nm, even with such lithography advances, circuit designers have to consider lithography-friendliness at design stage as part of design for manufacturability (DFM) [3].

Lithography hotspot detection (LHD) is such an essential task of DFM, which is no longer optional for modern sub-14nm VLSI designs. Lithography hotspot is a mask layout location that is susceptible to having fatal pinching or bridging owing to the poor printability of certain layout patterns. To avoid such unprintable patterns or layout regions, it is commonly required to conduct full mask lithography simulation to identify such hotspots. While lithography simulation remains as the most accurate method to recognize lithography hotspots, the procedure can be very time-consuming to obtain the full chip characteristics [4]. To speedup the procedure, pattern matching and machine learning techniques have been recently deployed in LHD to save the simulation time [5]–[7]. For example, [6] built a hotspot library to match and identify the hotspot candidates. Reference [7] extracted low-dimensional feature vectors from the layout clips and then employed machine learning or even deep learning techniques to predict the hotspots. Obviously, *the performance of all the aforementioned methods heavily depends on the quantity and quality of the underlying hotspot data to build the library or train the model*. Otherwise, these methods may have weak generality especially for unique design patterns or topologies under the advanced technology nodes.

In practice, each design houses may own a certain amount of hotspot data, which can be homogeneous[1] and possibly insufficient to build a general and robust model/library through *local learning*. On the other hand, the design houses are unwilling to directly share such data with other houses or even the tool developer to build one unified model through *centralized learning* due to privacy concern. Recently, advances in federated learning in the deep learning community provide a promising alternative to address the aforementioned dilemma. Unlike centralized learning that needs to collect the data at a centralized server or local training that can only utilize the design house's own data, *federated learning* allows each design house to train the model at local, and then uploads the updated model *instead of data* to a centralized server, which aggregates and re-distributes the updated global model back to each design house [8].

While federated learning naturally protects layout data privacy without direct access to local data, *its performance (or even convergence) actually can be very problematic when data are heterogeneous (or so-called non-Independent and Identically Distributed, i.e., non-IID)*. However, such heterogeneity is very common for lithography hotspot data, as each design house may have a very unique design pattern and layout topology, leading to lithography hotspot pattern heterogeneity. To overcome the challenge of heterogeneity in federated learning, the deep learning community recently introduced many variants of federated learning [9]–[12]. For example, federated transfer learning [9] ingested the knowledge from the source domain and reused the model in the target domain. In [10], the concept of federated multi-task learning is proposed to allow the model to learn the shared and unique features of different tasks. To provide more local model adaptability, [11]

*Xuezhong Lin and Jingyu Pan contribute equally to this work.

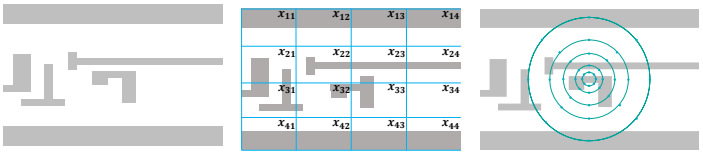
[1]Homogeneous hotspot data refers to the hotspot candidates that share the same feature space due to the similar design patterns or layout topologies.

used meta-learning to fine-tune the global model to generate different local models for different tasks. [13] further separated the global and local representations of the model through alternating model updates, which may get trapped at a sub-optimal solution when the global representation is much larger than the local one. A recent work [12] presented a framework called FedProx that added a proximal term to the objective to help handle the statistical heterogeneity. Note that LHD is different from the common deep learning applications: LHD is featured with limited design houses (several to tens) each of which usually has a reasonable amount of data (thousands to tens of thousands layout clips). The prior federated learning variants [9]–[13] are not designed for LHD and hence can be inefficient without such domain knowledge. For example, meta learning appears to loosely ensure the model consistency among the local nodes and hence *fails to learn the shared knowledge for LHD* when the number of local nodes is small, while FedProx strictly enforces the model consistency, yielding *limited local model adaptivity to support local hotspot data heterogeneity*. Thus, it is highly desired to have an LHD framework to properly balance local data heterogeneity and global model robustness.

To address the aforementioned issues in centralized learning, local learning, and federated learning, in this work, we propose an **accurate and efficient LHD framework using heterogeneous federated learning with local adaptation**. The major contributions are summarized as follows:

- The proposed framework accounts for the domain knowledge of LHD to design a heterogeneous federated learning framework for hotspot detection. A local adaptation scheme is employed to make the framework automatically balanced between local data heterogeneity and global model robustness.
- While many prior works empirically decide the low-dimensional representation of the layout clips, we propose an efficient feature selection method to automatically select the most critical features and remove unnecessary redundancy to build a more compact and accurate feature representation.
- A heterogeneous federated learning with local adaptation (HFL-LA) algorithm is presented to handle data heterogeneity with a global sub-model to learn shared knowledge and local sub-models to adapt to local data features. A synchronization scheme is also presented to support communication heterogeneity.
- We perform a detailed theoretical analysis to provide the convergence guarantee for our proposed HFL-LA algorithm and establish the relationship between design parameters and convergence performance.

Experimental results show that our proposed framework outperforms the other local learning, centralized learning, and federated learning methods for various metrics and settings on both open-source and industrial datasets. Compared with the federated learning and its variants [8], [12], the proposed framework can achieve 7-11% accuracy improvement with one order of magnitude smaller false positive rate. Moreover, our framework can maintain a consistent performance when the number of clients increases and/or the size of the dataset reduces, while the performance of local learning quickly degrades in such scenarios. Finally, with the guidance from the theoretical analysis, the proposed framework can achieve a faster convergence even with heterogeneous communication between the clients and central server, while the other methods take 5× iterations to converge.



(a) (b) (c)

Fig. 1: (a) An example of a layout clip; (b) Local density extraction; (c) Concentric circle sampling.

II. Background

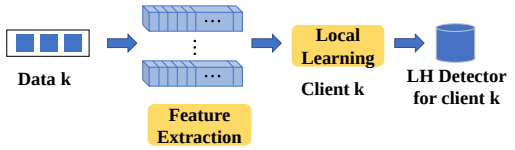
A. Feature Tensor Extraction

Feature tensor extraction is commonly used to reduce the complexity of high dimensional data. For LHD, the original data is hotspot and non-hotspot layout clips composed of polygonal patterns. Fig. 1(a) shows an example of a layout clip. If unprocessed layout clips are used as features in machine learning, the computational overhead would be huge. To address this issue, local density extraction and concentric circle sampling have been widely exploited in previous hotspot detection and optical proximity correction works [5], [14]. Fig. 1(b) shows an example of local density extraction that converts a layout clip to a vector. And Fig. 1(c) shows an example of concentric circle sampling which samples from the layout clip in a concentric circling manner. These feature extraction methods exploit prior knowledge of lithographic layout patterns, and hence can help reduce the layout representation complexity in LHD. However, *as the spatial information surrounding the polygonal patterns within the layout clip are ignored, such methods may suffer from accuracy issues* [5].

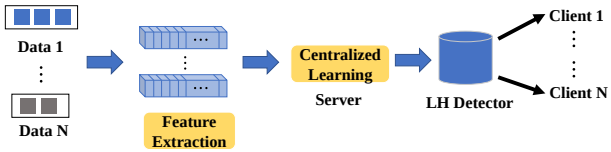
Another possible feature extraction is based on the spectral domain [5], [15], which can include more spatial information. For example, [5], [15] use discrete cosine transform (DCT) to convert the layout spatial information into the spectral domain, where the coefficients after the transform are considered as the feature representation of the clip. Since such feature tensor representation is still large in size and may cause non-trivial computational overhead, [15] proposes to ignore the high frequency components, which are supposed to be sparse and have limited useful information. However, such an assumption is not necessarily true for the advanced technologies, which can have subtle and abrupt changes in the shape. In other words, *the ignorance may neglect critical feature components and hence cause accuracy loss*.

B. Federated Learning

Federated learning allows local clients to collaboratively learn a shared model while keeping all the training data at local [8]. Consider a set of N clients connected to a central



(a) Procedure for local learning



(b) Procedure for centralized learning

Fig. 2: Two commonly used procedures for LHD.

TABLE I: Symbols used in the proposed framework.

Symbol	Definition
w	The set of weights of a CNN model
w_g	Global weights of the model
w_l^k	Local weights of the k_{th} client model
N	Total number of clients
n_k	The data size of client k

server, where each client can only access its own local data and has a local objective function $F_k : \mathbb{R}^d \to \mathbb{R}, k = 1, ..., N$. Federated learning can be then formulated as

$$\min_w f(w) = \frac{1}{N} \sum_{k=1}^{N} F_k(w), \tag{1}$$

where w is the model parameter, and f denotes the global objective function. FedAvg [8] is a popular federated learning method to solve the above problem. In FedAvg, the clients send updates of locally trained models to the central server in each round, 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.

III. Proposed Framework

A. Overview

Fig. 2 demonstrates two commonly used procedures for LHD, *i.e.*, local learning in Fig. 2(a) and centralized learning in Fig. 2(b). Both procedures contain two key steps, feature tensor extraction and learning. We adopt these two procedures as our baseline models for LHD. TABLE I defines the symbols that will be used in the rest of the paper.

The performance of LHD can be evaluated by the true positive rate (TPR), the false positive rate (FPR), and the overall accuracy, which can be defined as follows.

Definition 1 (True Positive Rate). *The ratio between the number of correctly identified layout hotspots and the total number of hotspots.*

Definition 2 (False Positive Rate). *The ratio between the number of wrongly identified layout hotspots (false alarms) and the total number of non-hotspots.*

Definition 3 (Accuracy). *The ratio between the number of correctly classified clips and the total number of clips.*

With the definitions above, we propose to formulate the following heterogeneous federated learning based LHD:

Problem Formulation 1 (Heterogeneous Federated Learning Based Lithography Hotspot Detection). *Given N clients (or design houses) owning unique layout data, the proposed LHD is to aggregate the information from the clients and create a compact local sub-model on each client and a global sub-model shared across the clients. The global and local sub-models form a unique hotspot detector for each client.*

The proposed heterogeneous federated learning based LHD aims to support the heterogeneity at different levels: data, model, and communication:

- **Data**: The hotspot patterns at each design house (client) can be non-IID.
- **Model**: The optimized detector model includes global and local sub-models, where the local sub-model can be different from client to client through the local adaptation.
- **Communication**: Unlike the prior federated learning [8], the framework allows asynchronous updates from the clients while maintaining good convergence.

Figure 3 presents an overview of the proposed framework to solve the above LHD problem with the desired features, which includes three key operations:

- **Feature Selection:** An efficient feature selection method is proposed to automatically find critical features of the layout clip and remove unnecessary redundancy.
- **Global Aggregation:** Global aggregation only updates the global sub-model shared across the clients with fewer parameters compared to the full model. It does not only reduces the computational cost but also facilitates heterogeneous communication.
- **Local Adaptation:** This operation allows the unique local sub-model at each client to have personalized feature representation of local non-IID layout data.

These operations connect central server and clients together to build a privacy-preserving system, which allows distilled knowledge sharing through federated learning and balance between global model robustness and local feature support. In the following, we will discuss the three operations in details.

B. Feature Selection

As discussed in Sec. II-B, while spectral based method can utilize more spatial information, it may easily generate a very large feature vector. To reduce computational cost, the vector is often shortened based on prior knowledge or heuristics [5], [15]. In this paper, we would like to propose a more automatic feature selection method to find out the most critical components while maintaining the accuracy.

The proposed selection procedure is shown in Fig. 4. The input layout clip is first mapped to a spectral domain with DCT. Then we use Group Lasso training to remove the unwanted redundancy [16], which is a common regularization

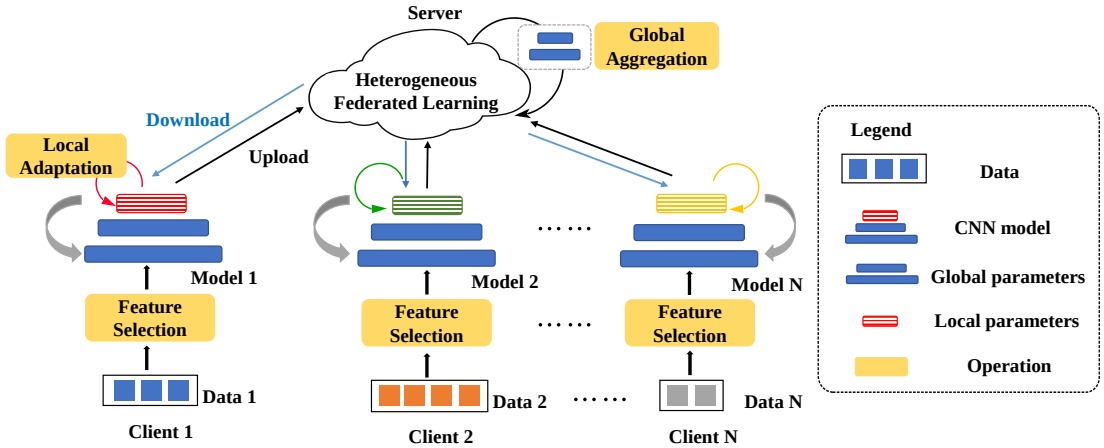


Fig. 3: Overview of the proposed LHD framework using heterogeneous federated learning with local adaptation.

to induce grouped sparsity in a deep CNN model. Generally, the optimization regularized by Group Lasso is

$$L(w) = L_D(w) + R(w) + \sum_{c=1}^{C} |R_{\ell_2}(w_c)|, \tag{2}$$

where w is the set of the weights, $L_D(w)$ is the loss on data, $R(w)$ is a general regularization term applied on all the weights ($e.g.$, L2-norm), and $R_{\ell_2}(w_c)$ is a structured L2 regularization on the specific c_{th} weight group w_c. In particular, if we make the channels of each filter in the first convolution layer of a deep CNN model a penalized group, the optimization would tend to remove less important channels. Since each channel directly corresponds to a channel in feature space, this is equivalent to removing the redundant feature channels. In other words, the remaining features are supposed to be the critical feature representation. The optimization target of the channel-wise Group Lasso penalty can be defined as

$$L(w) = L_D(w) + \lambda_R R(w) + \lambda_{\text{Lasso}} \sum_{c=1}^{C^{(0)}} \left|\left| w^{(0)}_{:,c,::} \right|\right|, \tag{3}$$

where $w^{(0)}$ is the weight of the first convolutional layer, $w^{(0)}_{:,c,:,:}$ is the c_{th} channel of all the filters in $w^{(0)}$, $R_{\ell_2}(w)$ is the L2 regularization term applied on all the weights, λ_R is the L2 regularization strength and λ_{Lasso} is the Group Lasso regularization strength. When c is a feature channel with less impact on the data loss, our feature selection method tends to enforce the L2 norm of all the weights related to the channel to zero. Then, the remaining feature channels would be the more critical features, leading to a reduction in the dimension of the layout clip information representation.

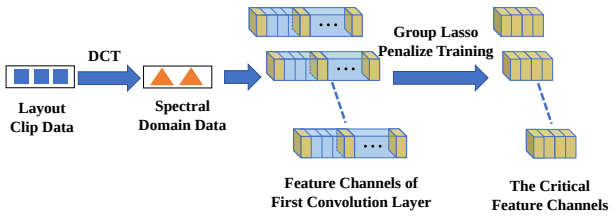


Fig. 4: Procedure of the proposed feature selection.

C. Global Aggregation and Local Adaptation

Global aggregation and local adaptation are the key operations in the proposed Heterogeneous Federated Learning with Local Adaptation algorithm (HFL-LA). The algorithm HFL-LA is particularly designed for LHD with awareness of its unique domain knowledge: (1) The design patterns in different clients (design houses) may have non-trivial portion in common, which indicates a larger global sub-model for knowledge sharing; (2) The number of clients may be limited, e.g., from several to tens; (3) The local data at each client may be insufficient to support a large local sub-model training.

As shown in Fig. 3, HFL-LA adopts a flow similar to the conventional federated learning that has a central server to aggregate the information uploaded from the distributed clients. However, unlike the conventional federated learning, the model that each client maintains can be further decomposed into a global sub-model and a local sub-model, where: (1) the global sub-model is downloaded from the server and shared across the clients to fuse the common knowledge for LHD, and (2) the local sub-model is maintained within the client to adapt to the non-IID local data and hence, varies from client to client.

To derive such a model, we define the following objective function for optimization:

$$\min_{w_g, w_l} \left\{ F\left(w_g, w_l\right) \triangleq \sum_{k=1}^{N} p_k F_k\left(w_g, w_l^k\right) \right\}, \tag{4}$$

where w_g is the global sub-model parameter shared by all the clients; $w_l := \left[w_l^1, \cdots, w_l^N\right]$ is a matrix whose k_{th} column is the local sub-model parameter for the k_{th} client; N is the number of clients; $p_k \geqslant 0$ and $\sum_{k=1}^{N} p_k = 1$ is the contribution ratio of each client; n_k is the data size of client k. By default, we can set $p_k = \frac{n_k}{n}$, where $n = \sum_{k=1}^{N} n_k$ is the total number of samples across all the clients. For the local data at client k, $F_k\left(\cdot\right)$ is the local (potentially non-convex) loss function, which is defined as

$$F_k\left(w_g, w_l^k\right) = \frac{1}{n_k} \sum_{j=1}^{n_k} \ell\left(w_g, w_l^k; x_{k,j}\right), \tag{5}$$

where $x_{k,j}$ is the j_{th} sample of client k. As shown in

Algorithm 1, in the t round, the central server broadcasts the latest global sub-model parameter $w_{t,g}$ to all the clients. Then, each client ($e.g.$, k_{th} client) starts with $w_t^k = w_{t,g} \cup w_{t,l}^k$ and conducts E_l ($\geqslant 1$) local updates for sub-model parameters

$$w_{t+\frac{1}{2},l}^k = w_{t,l}^k - \eta \sum_{i=0}^{E_l-1} \nabla_l F_k \left(w_{t,g}, \hat{w}_{t+i,l}^k; \xi_t^k\right), \tag{6}$$

where $\hat{w}_{t+i,l}^k$ denote the intermediate variables locally updated by client k in the t round; $\hat{w}_{t,l}^k = w_{t,l}^k$; ξ_t^k are the samples uniformly chosen from the local data in the t round of training. After that, the global and local sub-model parameters at client k become $w_{t+\frac{1}{2}}^k = w_{t,g} \cup w_{t+\frac{1}{2},l}^k$ and are then updated by E steps of inner gradient descent as follows:

$$w_{t+1}^k = w_{t+\frac{1}{2}}^k - \eta \sum_{i=0}^{E-1} \nabla F_k \left(\hat{w}_{t+\frac{1}{2}+i}^k; \xi_t^k\right), \tag{7}$$

where $\hat{w}_{t+\frac{1}{2}+i}^k$ denote the intermediate variables updated by client k in the $t+\frac{1}{2}$ round; $\hat{w}_{t+\frac{1}{2}}^k = w_{t+\frac{1}{2}}^k$. Finally, the client sends the global sub-model parameters back to the server, which then aggregates the global sub-model parameters of all the clients, $i.e.$, $w_{t+1,g}^1, \cdots, w_{t+1,g}^N$, to generate the new global sub-model, $w_{t+1,g}$.

Algorithm 1 HFL-LA algorithm for LHD

Server:

Initialize $w_{0,g}$, send $w_{0,g}$ to every client;
for each round $t = 0, 1, \cdots, T-1$ **do**
  $S_t \leftarrow$ (Randomly select K cilents);
  for each client $k \in S_t$ **do**
    $w_{t+1,g}^k \leftarrow$ ClientUpdate(k, $w_{t,g}$);
  $w_{t+1,g} \leftarrow \sum_{k=1}^K \frac{n_k}{n_K} w_{t+1,g}^k$;
  Send $w_{t+1,g}$ to every client.

Client:

ClientUpdate(k, w_g):
$\mathcal{B} \longleftarrow$ (Divide $\mathcal{D}_k$ according to the batch size of B);
for each local update $i = 0, 1 \cdots, E_l$ **do**
  for batch $\xi^k \in \mathcal{B}$ **do**
    $w_l^k \leftarrow w_l^k - \eta \nabla_l F_k(w_l^k; \xi^k)$;
for each global update $i = 0, 1 \cdots, E$ **do**
  for batch $\xi^k \in \mathcal{B}$ **do**
    $w_g^k \cup w_l^k \leftarrow w_g \cup w_l^k - \eta \nabla F_k(w_g \cup w_l^k; \xi^k)$;
return w_g^k to server.

Fig. 5 presents the network architecture of each client used in our experiment. The architecture has two convolution stages and two fully connected stages. Each convolution stage has two convolution layers, a Rectified Linear Unit (ReLU) layer, and a max-pooling layer. The second fully connected layer is the output layer of the network in which the outputs correspond to the predicted probabilities of hotspot and non-hotspot. We note that the presented network architecture is just a specific example for the target application and our proposed framework is not limited by specific network architectures.

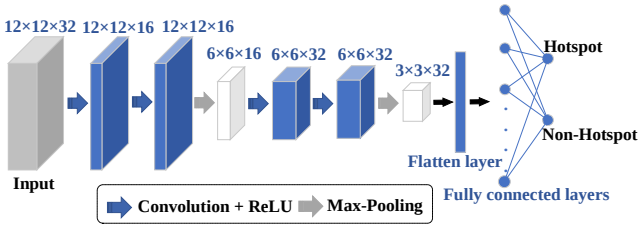


Fig. 5: Neural network architecture example at the client.

D. Communication Heterogeneity

In addition to data heterogeneity, the proposed framework also supports communication heterogeneity, $i.e.$, the clients can conduct synchronized or asynchronized updates, while still ensuring good convergence. For the synchronized updates, all the clients participate in each round of global aggregation as:

$$w_{t+1,g} = \sum_{k=1}^{N} p_k w_{t+1,g}^k. \tag{8}$$

Then all the clients need to wait for the slowest client to finish the update. Due to heterogeneity of data, the computational complexity and willingness to participate in a synchronized or asynchronized update may vary from client to client. Thus, it is more realistic to assume that different clients may update at different rates. We can set a threshold K ($1 \leqslant K < N$) and let the central server collect the outputs of only the first K responded clients. After collecting K outputs, the server stops waiting for the rest clients, $i.e.$, the $(K+1)_{th}$ to N_{th} clients are ignored in this round of global aggregation. Assuming S_t ($|S_t| = K$) is the set of the indices of the first K clients in the t_{th} round, the global aggregation can then be rewritten as

$$w_{t+1,g} = \frac{n}{n_K} \sum_{k \in S_t} p_k w_{t+1,g}^k, \tag{9}$$

where n_K is the sum of the sample data volume of the first K clients and $\frac{n}{n_K} \sum_{k \in S_t} p_k = 1$.

IV. Convergence Analysis

In this section, we study the convergence of the proposed HFL-LA algorithm. Unlike the conventional federated learning, our proposed HFL-LA algorithm for LHD works with fewer clients, smaller data volume, and non-IID datasets, making the convergence analysis more challenging. Before proceeding into the main convergence result, we provide the following widely used assumptions on the local cost functions $\{F_k\}$ and stochastic gradients [17].

Assumption 1. *$F_1, \cdots, F_N$ are all L-smooth, i.e., $\forall v, w$, $\|\nabla F_k(v) - \nabla F_k(w)\| \leqslant L\|v - w\|, \forall k = 1, ..., N$.*

Assumption 2. *Let ξ_i^k be uniformly sampled from the k_{th} client's local data. The variance of stochastic gradients in each client is upper bounded, i.e., $\mathbb{E}\left\|\nabla F_k\left(w_i^k; \xi_i^k\right) - \nabla F_k\left(w_i^k\right)\right\|^2 \leqslant \sigma^2$.*

Assumption 3. *The expected squared norm of stochastic gradients is uniformly bounded by a constant G^2, i.e., $\mathbb{E}\left\|\nabla F_k\left(w_i^k; \xi_i^k\right)\right\|^2 \leqslant G^2$ for all $k = 1, \cdots, N$.*

TABLE II: Details of the benchmarks, ICCAD and Industry.

Benchmarks	Training Set		Testing Set		Size/Clip (μm^2)
	HS#	non-HS#	HS#	non-HS#	
ICCAD	1204	17096	2524	13503	3.6×3.6
Industry	3629	80299	942	20412	1.2×1.2

With the above assumptions, we are ready to present the following main results of the convergence of the proposed algorithm. *The detailed proof can be found in the Appendix.*

Lemma 1 (Consensus of global sub-model parameters)**.** *Suppose Assumption 3 holds. Then,*

$$\mathbb{E}\left\|\frac{1}{N}\sum_{j=1}^{N} w_{t,g}^{j} - w_{t,g}^{k}\right\|^{2} \leqslant \eta^{2}\left(E-1\right)^{2} G^{2}. \tag{10}$$

The above lemma guarantees that the global sub-model parameters of all the clients reach consensus with an error proportional to the learning rate η while the following theorem ensures the convergence of the proposed algorithm.

Theorem 1. *Suppose Assumption 1-3 hold. Then, $\forall T > 1$, we have*

$$\begin{aligned}&\frac{1}{T}\sum_{t=0}^{T-1}\frac{1}{N}\sum_{k=1}^{N}\left[\left\|\nabla F_{k}\left(w_{t}^{k}\right)\right\|^{2}\right]\\ &\leqslant \frac{2\left[\frac{1}{N}\sum_{k=1}^{N} F_{k}\left(w_{0}^{k}\right) - F^{*}\right]}{T\eta}\\ &+ \mathcal{O}\left(\eta L G^{2}\right) + 2\sqrt{N}\left(E-1\right)G\left(\sigma^{2}+G^{2}\right).\end{aligned} \tag{11}$$

Remark 1. *The above theorem shows that, with a constant step-size, the parameters of all clients converge to the η-neighborhood of a stationary point with a rate of $\mathcal{O}\left(1/T\right)$. It should be noted that the second term of the steady-state error is proportional to the square root of N, but will vanish when $E = 1$. This theorem sheds light on the relationship between design parameters and convergence performance, which helps guide the design of the proposed HFL-LA algorithm.*

V. Experimental Results

We implement the proposed framework using the PyTorch library [18]. We use the following hyperparameters to conduct model training on each client in our experiment: We train our models with Adam optimizer for $T = 50$ rounds with a fixed learning rate $\eta = 0.001$ and a batch size of 64. And in each round, we conduct local updates for $E_l = 500$ iterations, and global updates for $E = 1500$ iterations. To prevent overfitting, we use L2 regularization of 0.00001. We adopt two benchmarks (`ICCAD` and `Industry`) for training and testing. We merge all the 28nm patterns in the test cases published in ICCAD 2012 contest [19] into a unified benchmark denoted by `ICCAD`. And `Industry` is obtained from our industrial partner at 20nm technology node. Table II summarizes the benchmark details including the training/testing as well as the layout clip size. In the table, columns "HS#" and "non-HS#" list the total numbers of hotspots and non-hotspots, respectively. Since the original layout clips have different sizes, clips in `ICCAD` are divided into nine blocks to have a consistent size as `Industry`. We note that, due to the different technologies and design patterns, the two benchmarks have different feature representations, and `Industry` has more diverse design patterns (*i.e.*, higher data heterogeneity) than `ICCAD`.

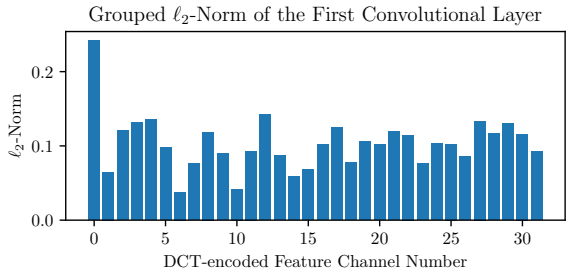


Fig. 6: Grouped ℓ_2-norm of the first convolution layer. The DCT-encoded channel number spans from 0 to 31, where the channel 0 denotes the DC component of DCT of the layout clip, and the channels 1-31 denote the AC components of different frequencies.

A. Feature Selection

This subsection presents the performance of the proposed feature selection method. As discussed in Sec. III-B, L2 norm of the channel-wise groups in the first convolutional layer is correlated with the contributions to model performance from the corresponding feature channels, as shown in Fig. 6. We then sort all the feature channels by their L2 norms and retrain our model from scratch with the selected top-k channels, *i.e.*, $k = 26$ in the experiment. To validate the efficiency of our feature selection method, we test the performance of HFL-LA with different numbers of features representing the layout clips on the validation set and compare the performance. Fig. 7 shows that HFL-LA achieves comparable (even slightly higher) accuracy in the case of $k = 26$ features as suggested by the proposed selection method for both benchmarks, which indicates a 18.75% computation reduction for the following learning in comparison to the original 32 features.

B. Heterogeneous Federated Learning with Local Adaptation

To demonstrate the performance of the proposed HFL-LA algorithm, we compare the results of HFL-LA with that of

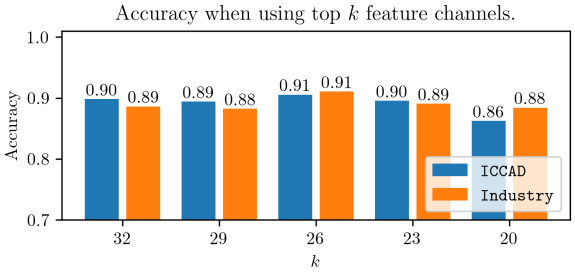


Fig. 7: Accuracy of HFL-LA on the validation set using different number of features representing the layout clip.

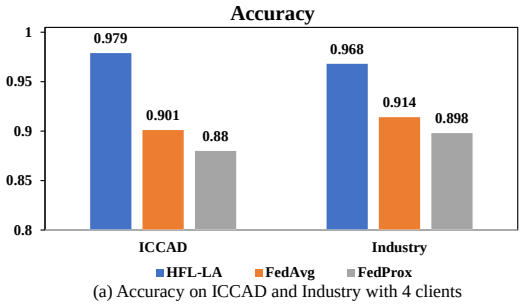
TABLE III: Inference performance (TPR, FPR and accuracy) comparison among HFL-LA, FedAvg, FedProx, local and central learning.

Methods	Number of clients	ICCAD			Industry		
		TPR	FPR	ACC	TPR	FPR	ACC
HFL-LA	2 clients	0.960	0.019	**0.980**	0.966	0.040	0.964
	4 clients	0.967	0.021	**0.979**	0.975	0.049	**0.968**
	10 clients	0.967	0.030	0.970	0.971	0.050	**0.965**
FedAvg	2 clients	0.974	0.110	0.892	0.814	0.010	0.869
	4 clients	0.971	0.101	0.901	0.883	0.016	0.914
	10 clients	0.969	0.090	0.911	0.881	0.016	0.913
FedProx	2 clients	0.977	0.134	0.868	0.854	0.014	0.895
	4 clients	0.973	0.121	0.880	0.859	0.017	0.898
	10 clients	0.958	0.113	0.888	0.843	0.016	0.887
Local	2 clients	0.973	0.021	0.978	0.976	0.039	**0.971**
	4 clients	0.966	0.021	0.978	0.971	0.071	0.957
	10 clients	0.925	0.024	**0.975**	0.954	0.123	0.930
Centralized	1 server	0.956	0.032	0.968	0.974	0.038	0.970

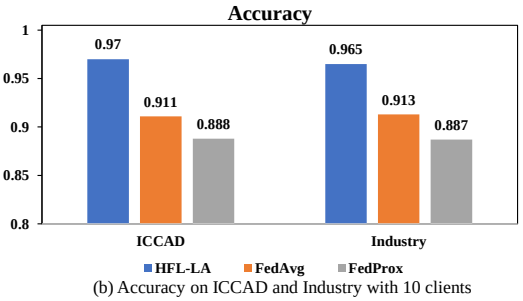
the state-of-the-art federated learning algorithm, FedAvg in [8] and FedProx in [12], as well as local and central learning. Here we have:

- FedAvg: The conventional federated learning algorithm that averages over the uploaded model [8].
- FedProx: The algorithm adds a proximal term to the objective to handle the heterogeneity [12].
- Local learning (denoted as local): The local learning algorithm that only uses the local data of client.
- Central learning (denoted as centralized): The central learning algorithm has access to all the training sets to train one unified model.

In our experiments, the training sets of `ICCAD` and `Industry` benchmarks are merged together and then assigned to different numbers of clients as the local data, *i.e.*, 2, 4, and 10 clients. The testing sets have been preserved in advance as shown in Table II and used to validate the performance of the trained models. We compare the performance of the algorithms in terms of TPR, FPR, and accuracy, as defined in Sec. III-A, and summarize the results in Table III. In the experiments in Table III, all the clients communicate with the server in a synchronous manner and the average of the performance across all the clients for the three scenarios of 2, 4, and 10 clients, in which the best performance cases are marked in bold. It is noted that the proposed HFL-LA can achieve 7-11% accuracy improvement for both TPR and FPR, compared to FedAvg and FedProx. Due to the fact of using only local homogeneous training data, local learning can achieve slightly better results for `ICCAD`. However, when the data heterogeneity increases like `Industry`, the performance of local learning quickly drops and yields ∼4% degradation compared to HFL-LA.

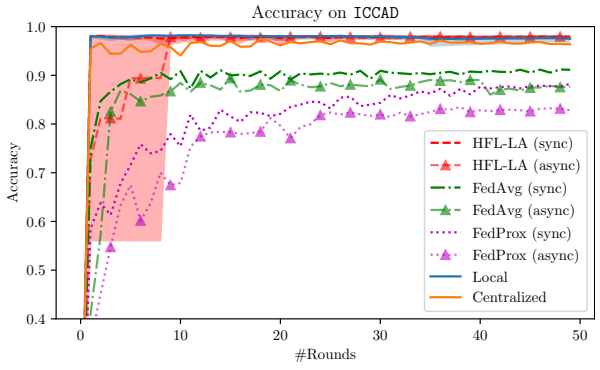


(a) Accuracy on ICCAD and Industry with 4 clients

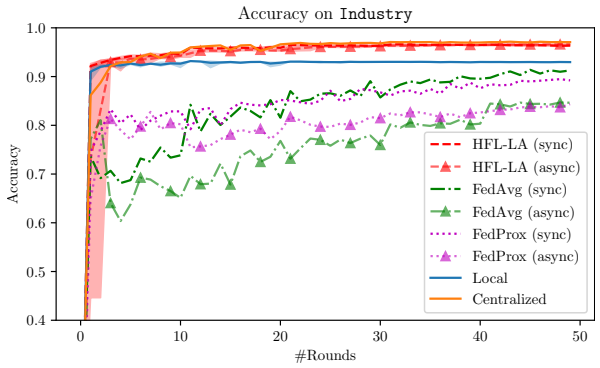


(b) Accuracy on ICCAD and Industry with 10 clients

Fig. 8: Accuracy comparison among HFL-LA, FedAvg, and FedProx on `ICCAD` and `Industry` with 4 and 10 clients using asynchronous model updates.



(a) Accuracy on ICCAD testing set



(b) Accuracy on Industry testing set

Fig. 9: Convergence comparison among the different methods on `ICCAD` and `Industry` during training with the models evaluated on the testing sets.

We further compare the results when the model can be updated asynchronously for the scenarios of 4 and 10 clients, where half of the clients are randomly selected for training and update in each round. Since only federated learning based methods require model updates, we only compare HFL-LA with FedAvg and FedProx in Fig. 8. As shown in the figure, even with heterogeneous communication and updates, HFL-LA can still achieve 5-10% accuracy improvement from that

of the other federated learning methods [8], [12].

Finally, we compare the accuracy changes of different methods with different update mechanisms (synchronous and asynchronous, denoted as sync and async, respectively) for 10 clients during the training. For `ICCAD` benchmark in Fig. 9(a), local learning and HFL-LA method achieve the highest accuracy and converge much faster than the other methods. Even with asynchronous updates, HFL-LA method can achieve convergence rate and accuracy similar to the synchronous case. For `Industry` in Fig. 9(b), the superiority of HFL-LA is more obvious, outperforming all the other methods in terms of accuracy (e.g., 3.7% improvement over local learning). Moreover, HFL-LA achieves almost $5\times$ convergence speedup compared to the other federated learning methods even adopting asynchronous updates.

VI. Conclusion

In this paper, we propose a novel heterogeneous federated learning based hotspot detection framework with local adaptation. By adopting an efficient feature selection and utilizing the domain knowledge of LHD, our framework can support the heterogeneity in data, model, and communication. Experimental results shows that our framework not only outperforms other alternative methods in terms of performance but can also guarantee good convergence even in the scenario with high heterogeneity.

References

[1] G. E. Moore *et al.*, "Cramming more components onto integrated circuits," 1965.

[2] T. Matsunawa, Y. Bei, and D. Z. Pan, "Optical proximity correction with hierarchical bayes model," in *Optical Microlithography XXVIII*, 2015.

[3] Y. T. Yu, Y. C. Chan, S. Sinha, H. R. Jiang, and C. Chiang, "Accurate process-hotspot detection using critical design rule extraction," *ACM*, 2012.

[4] J. Kim and M. Fan, "Hotspot detection on post-opc layout using full chip simulation based verification tool: A case study with aerial image simulation," *Proceedings of SPIE - The International Society for Optical Engineering*, 2003.

[5] H. Yang, S. Jing, Z. Yi, Y. Ma, and E. Young, "Layout hotspot detection with feature tensor generation and deep biased learning," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. PP, no. 99, pp. 1–1, 2017.

[6] W. Wen, J. Li, S. Lin, J. Chen, and S. Chang, "A fuzzy-matching model with grid reduction for lithography hotspot detection," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 33, no. 11, pp. 1671–1680, 2014.

[7] Y. Yu, G. Lin, I. H. Jiang, and C. Chiang, "Machine-learning-based hotspot detection using topological classification and critical feature extraction," in *IEEE*, 2015, pp. 460–470.

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

[9] Y. Liu, Y. Kang, C. Xing, T. Chen, and Q. Yang, "A secure federated transfer learning framework," *Intelligent Systems, IEEE*, vol. PP, no. 99, pp. 1–1, 2020.

[10] V. Smith, C.-K. Chiang, M. Sanjabi, and A. Talwalkar, "Federated multi-task learning," 2018.

[11] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," 2017.

[12] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," 2018.

[13] P. P. Liang, T. Liu, Z. Liu, R. Salakhutdinov, and L. P. Morency, "Think locally, act globally: Federated learning with local and global representations," 2020.

[14] T. Matsunawa, B. Yu, and D. Z. Pan, "Optical proximity correction with hierarchical bayes model," in *Optical Microlithography XXVIII*, vol. 9426. International Society for Optics and Photonics, 2015, p. 94260X.

[15] H. Yang, J. Su, Y. Zou, Y. Ma, B. Yu, and E. F. Young, "Layout hotspot detection with feature tensor generation and deep biased learning," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 38, no. 6, pp. 1175–1187, 2018.

[16] M. Yuan and Y. Lin, "Model selection and estimation in regression with grouped variables," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 68, no. 1, pp. 49–67, 2006.

[17] H. Yu, S. Yang, and S. Zhu, "Parallel restarted sgd with faster convergence and less communication: Demystifying why model averaging works for deep learning," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 5693–5700, 2019.

[18] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *arXiv preprint arXiv:1912.01703*, 2019.

[19] J. A. Torres, "Iccad-2012 cad contest in fuzzy pattern matching for physical verification and benchmark suite," in *2012 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 2012, pp. 349–350.