Model-Agnostic Federated Learning for Privacy-Preserving Systems

Hussain M. J. Almohri*, Layne T. Watson†
*Department of Computer Science, Kuwait University, Kuwait
Email: almohri@cs.ku.edu.kw
†Departments of Computer Science, Mathematics, and Aerospace and Ocean Engineering,
Virginia Polytechnic Institute & State University, Blacksburg, VA 24061
Email: ltw@ieee.org

***Abstract*—This study presents an innovative aggregation scheme for model-agnostic, local, heterogeneous data models within the domain of Federated Learning. The proposed approach imposes minimal constraints on local models, only necessitating local model parameters and distances from local data centroids for a particular query. These requirements facilitate the design of privacy-preserving learning systems. We introduce a system architecture based on federated interpolation to operationalize the proposed scheme. The accuracy of our proposed scheme is evaluated using two distinct real-world datasets. We compare our results to the extreme case of a single-client scenario having complete access to all data points. Our findings indicate that, on average, federated interpolation maintains robust accuracy, experiencing a slight reduction of less than 10% compared to the single-client model with full data access.**

***Index Terms*—Distributed Artificial Intelligence, Security and Privacy Protection, Interpolation**

I. Introduction

Federated machine learning [1] is an emerging machine learning paradigm to learn from data distributed among distrusting parties without revealing the raw data. In federated learning (FL), the learning process is managed by a federated learning server, which coordinates the collaboration of multiple clients who possess local datasets. Rather than exposing their sensitive data to each other or to the server, the clients participate in iterations of building a central model at a server. Clients can be requested to update the parameters of a central model by executing one or several iterations of the stochastic gradient descent algorithm that aims to minimize the model's error according to a client's local dataset. As opposed to differential privacy [2] (where data is modified to provide privacy), federated learning protects the data privacy by minimizing the exposure of the data possessed by the clients yet building a useful model of the data.

Federated learning presents multiple challenges. At its core, FL involves many interactions between clients and a central server to create a unified model from complementary client datasets. To protect the clients' datasets, a federated learning system must minimize the exposure of client data. The reason is that the central aggregating server (often a cloud service) might attempt to exploit and misuse the client datasets. Moreover, it is crucial to protect client model parameters from data recovery attacks as highlighted by Zhang et al. [3].

FL requires frequent client and server communications to establish and refine the central model. The high communication requirement results from executing the iteration of the gradient descent algorithm. While some methods have been proposed to reduce this communication [4], regular interactions remain a foundational aspect of common FL approaches.

Lastly, while FL is well-established for image classification and natural language processing [5], there's uncertainty about its effectiveness with tabular datasets, despite the existing demand [6].

In this work, we introduce the *federated model interpolation* scheme and a federated learning architecture (Figure 1) that allow building a central model with no data or model parameter sharing. Federated model interpolation is based on Shepard's original algorithm [7], [8], which is used for high-dimensional numerical interpolation. Federated model interpolation modifies Shepard's algorithm to interpolate client models, instead of values of a function of multiple dimensions.

In our scheme, each client chooses and builds a model from its dataset for predicting future values. Clients aim to collaborate and build a central model. A trusted but curious server builds an interpolant by asking each client to provide a centroid (such as geometric average) that summarizes the client's dataset. Upon receiving a query for predicting future values, the interpolant (an equation involving client centroids) effectively selects the best client or combination of clients to produce the predicted value corresponding to the query. The selected clients produce their response values (instead of sharing their model parameters) and the interpolant produces the final predicted value according to the select client weights.

Previous research has presented various model aggregation strategies for federated learning, extending the initial federated averaging concept for aggregating gradient descent iterations from clients [1]. Some notable techniques include model interpolation [9], meta model formation [10], and model fusion [10]. Specifically, model interpolation, closest to our work, allows interpolation without revealing the full dataset. However, model interpolation presented by Mansour et al.'s demands substantial data from clients to fine-tune its interpolation components. Our approach offers a privacy-centric, model-agnostic learning system where clients don't need to disclose any data subset.

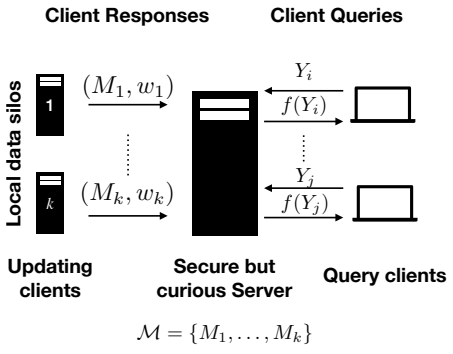

Fig. 1. Federated interpolation process. A query invokes calls to clients, requesting for each client's response. The responses are aggregated in the server and a unique response is sent to the querying client.

Our work is distinguished from the state of the art in three ways. First, federated model interpolation avoids sharing raw data or model parameters. While federated interpolation does not garble the data (as in the case with differential privacy [11]), it helps protecting client data by only querying their models. Second, federated interpolation requires a single communication round for building the model. When responding to a query, another round of communication is required for only a selected subset of clients with models that are more likely to suit the query. Third, tabular data does not exhibit numerically clear and distinct patterns as in image classification. Moreover, clients may want to use different local models to fit their datasets. Thus, federated model interpolation works with tabular datasets since it redirects the query towards the clients that have datasets that are numerically closer to the submitted query.

The core contribution of this work is a new federated model interpolation scheme for federated classification tasks of tabular datasets, utilizing Shepard's original interpolation scheme. This paper also presents

1) the design of a data model, a learning process, and a federated learning system benefiting from federated model interpolation (Section III),
2) the details of federated model interpolation given noisy datasets with the objective of binary or multilabel classification (Section IV), and
3) experiments on real-world datasets to compare the classification accuracy of individual local models with that of a global model (Section V).

To evaluate the proposed federated model interpolation, we used two real-world datasets: a COVID-19 metadata dataset [12] and the KDD CUP 99 [13] network requests dataset. In the COVID-19 metadata dataset the learning task is binary, predicting whether a patient survives. In KDD CUP 99, the learning task is multilabel classification in which the type of communication is classified. Per our experiments, compared to a single-client scenario, federated interpolation has an average $\approx 7\%$ reduction in accuracy with the COVID-19 metadata dataset and an average $\approx$ 10% reduction in accuracy with the KDD CUP 99 dataset. As shown later in Section V, in some cases there is no accuracy loss due to using federated interpolation.

II. Related Works

Some of the previous work uses the central server to produce a unified global model, only delegating the gradient descent algorithm iterations to the participating clients(e.g., [14], [15]). Others elect only to perform model aggregation (e.g.,[9]), leaving the learning task to the participating clients.

FedProto, by Tan et al. [16], is a federated interpolation approach similar to our work. FedProto uses prototype-based learning to maintain the privacy of clients, operates under the assumption of cross-client federated learning, and applies to image classification problems. Our proposed scheme does not use simple averaging. Instead, each client is asked to produce a local model value for a given query and a norm distance of the query from the centroid of the local dataset.

Mansour et al. developed three approaches to federated learning, including the idea of *model interpolation* [9] implicit in the earlier work of Shepard. In general, the proposed method requires the optimization of the unified model using data from clients, so the primary promise of preserving data privacy is defeated. Our federated interpolation (Section IV) scheme moves beyond simple model interpolation and eliminates all client participation in building the unified global model except for providing the hyperparameters of the local models and centroid values representing the data collected by participating clients.

Yurochkin et al. introduce a novel beta Bernoulli process through which a *meta model* of optimized local models is produced [10]. Our model is simpler and faster to compute and makes no particular assumption on the parameters of local models. The advantage of using interpolation to aggregate local models is that federated interpolation is model agnostic, and does not require wholly disjoint or independent datasets.

Similar to the work by Yurochkin et al. [10], Claici et al. proposed to minimize KullbackLeibler (KL) divergence for *fusing* multiple heterogeneous local models, aiming to recover a mean-field approximation to the posterior distribution [17]. Their proposed scheme results in averaging the parameters of models produced by participating clients. The work in [17] moves a step towards our goals. However, with federated interpolation, no minimization is necessary at the server-side.

III. Security Model

In a federated learning system, a client-server architecture involving a coordinating server and multiple clients contributes to learning a global data model. Clients of the system can also query the global data model to predict future events. In contrast, distributed learning requires each client to execute iterations of the learning algorithm, thus using a central data silo stored at the server. Distributed learning improves computation time but does not maintain the privacy of client data.

Ensuring an accurate federation of client models requires a secure environment, trusting the integrity of the dataset D_i for

$1 \leq i \leq n$. The main concern is to maintain the confidentiality of each D_i, minimizing the privacy loss by each client $C_i \in C = \{C_1, \ldots, C_n\}$.

All data flow among any pair of (participating) clients $(C_i, C_j) \in C \times C$ and between any C_i and the server S is trusted. That is, each participant C_i can authenticate another participant C_j or the server S when needed. Participants are responsible for maintaining integrity and confidentiality of data on their local storage. The communication (through an `update(·)` or a `query(·)`) of any subset of interacting participants is secured using standard protocols such as Transport Layer Security (TLS). The considered security model is practical when several organizations collaborate to produce a global model.

A central authority can authenticate all participating clients. Client authentication does not necessarily imply correctness of a local client dataset. A client could have been compromised and the local dataset could have been maliciously altered. However, this threat model is beyond the scope of the current work.

The server may be curious and may inadvertently not maintain the privacy of any data that it receives. The server may attempt to benefit (e.g., financially) from client data or provide the data to authorities when required. Thus, a client C_i should not be required to share any subset of the dataset D_i with the server S or with any other client. Further, client participation in the federated learning protocol should not lead to inferring useful information.

IV. Federated Interpolation

This section presents the mathematical construction of the federated interpolation method (IV-A), followed by the design of a system to utilize the presented model (IV-B).

A. Construction of Federated Interpolant

Data points are $\big(x, f(x)\big)$ where $x \in E^p$ is a real p-dimensional vector and the function $f(x) \in E^m$ is a real m-dimensional vector. The ith client has data $\big\{\big(x_{ij}, f(x_{ij})\big)\big\}_{j \in I_i}$ for which the x_{ij} form a point cloud around the centroid

$$\bar{x}_i = (1/|I_i|) \sum_{j \in I_i} x_{ij}.$$

These point clouds can overlap (meaning their convex hulls intersect), but ideally, the centroids are well-separated. For example, one point cloud could provide pandemic mortality data for a specific age group and another point cloud from a different client provide data for a distinct age group with a slight overlap.

Each client i has a local model $M_i(x) \approx f(x)$ built from that client's data using common learning algorithms. The problem is given an argument Y, use the client models M_i to estimate $f(Y)$. The solution is realized by producing an interpolation of these local models M_i around the given argument Y, calculating the weights of individual models in producing the response $f(Y)$. Assume that the data $f(x)$ is noisy, so that the trivial solution of finding the x_{ij} closest to Y (each client i returns the distance $\|x_{ik} - Y\| = \min_j \|x_{ij} - Y\|$ and $f(x_{ik})$, or the average in case of a tie for the minimum distance) and then using $f(Y) \approx f(x_{ij})$ for this closest x_{ij} is not adequate.

For noisy data, the client models $M_i(x)$ are more robust, so the approach is to use Shepard's algorithm to interpolate the models. Precisely, given a query $Y \in E^p$, define

$$w_i = \frac{1}{\|Y - \bar{x}_i\|},$$

where $\|\cdot\|$ is the s-norm for some s (1, 2, 3, ∞, e.g.), and then approximate the response at Y by

$$f(Y) \approx \sum_i \left(\frac{w_i}{\sum_j w_j} \right) M_i(Y), \tag{1}$$

Federated interpolation balances the participation of models with centroids closer to the provided argument Y. Models with farther centroids receive lower weights and thus have less opportunity to influence the value $f(Y)$. This model can produce improved output from one that is generated by simply averaging over individual client models (as in [1]).

B. Design of the Interpolation System

Using the federated interpolation, we present the design of a privacy-preserving system for learning constructing a model from the datasets supplied by a federated set of clients.

Components and Privileges. The architecture of the federated interpolation of local client models is in Figure 1. In this architecture, the system has a central server and several clients as participants. The system distinguishes between updating clients $C^u \subset C$ and query clients $C^q \subset C$ in two different privilege classes. An updating client privilege also implies the query privilege. The server is in a separate privilege class and is the only participant that can respond to queries. The server receives infrequent updates from participating clients.

The architecture has two states: (i) bootstrapping and (ii) learning. The bootstrapping state is where the server registers and authenticates clients. Next, when in a learning state, the system receives client queries and coordinates generating an interpolated response. The system can start responding to query clients with as little as a single model from participating clients. The registration typically requires client verification and generation of cryptographic keys for future authentication. We do not address the details of client registration, which can follow standard registration protocols.

The Interpolation Process. Query clients start querying the server by sending specific query messages asynchronously. Algorithm 1 specifies the steps for processing a single query. The parameters of the algorithms are the set of query clients C^q, the set of models $\mathcal{M}$, the set of centroids $\mathcal{X}$, the norm parameter s for the weights w_i, and ϵ that controls the balance of weights in Lines 14–16. Algorithm 1 makes a call to Algorithm 2 to receive the clients' inputs. A call to Algorithm 2 is referred to as `local(`Y`)`.

Algorithm 1 starts by checking if at least a single local model exists (indicating that a minimum of a single model participating client is required). The main outer loop is responsible for receiving and processing requests. The `query(·)` (Section III) is used to receive query requests asynchronously (indicated by `await`). Once the query is received, the client privilege class is verified using `verify (·)`. Responding to the query Y requires building an interpolant based on the construction in Section IV-A. The first for-loop (Lines 8–11) is responsible for computing the weights and their sum. Next, Algorithm 1 computes the average of the weights to decide if the weights are *conclusive*. That is, we introduce a heuristic to avoid a condition where all weights are nearly equal. In this case, the algorithm cannot conclude which model is closest to the submitted query, and the computed response $f(Y)$ can be inaccurate. Thus, the second for-loop (Lines 13–17) removes models with an ϵ deviation from the mean, leaving the maximum weight in place. The final for-loop (Lines 18–20) computes the response r (which corresponds to $f(Y)$), which is returned in Line 24.

Algorithm 1 requires $\mathcal{O}(n\alpha)$ operations, with n clients/models and α operations for performing a prediction per model. Note that the call to a model $M_i(Y)$ is dependent on the algorithm used to train $M_i(Y)$.

From each client i with data centroid $\bar{X}_i$, the server needs to know the single number $\|X-\bar{X}_i\|_p$ in order to calculate the weights used in Shepard interpolation. If the query X is high-dimensional and real (or rational) valued, this single (nonzero) norm number cannot be used to infer either the centroid $\bar{X}_i$ or the client's data points defining the centroid. Even in the rare event that $X = \bar{X}_i$, the server cannot infer with certainty any client i data point from the centroid.

Algorithm 2 assists Algorithm 1 by retrieving client update responses. Each client sends the response of its local model and the computed weight according to the submitted query to the server. The server broadcasts an asynchronous request (using the `async` keyword) to the k registered clients in C^u. Each client in C^u has the update privilege and is assumed to have generated a model M_i (fit to the local dataset) and a centroid $\bar{x}_i$ from the dataset D_i. The server will *not* receive any information on model parameters or the local data centroids. Instead, the client computes the response of its local model $M_i(Y)$ and its model weight. The `async update` call awaits the computed values from each client. All the computed values are stored in $\mathcal{M}$ and $\mathcal{X}$ and returned.

Algorithm 1 Federated Query Processing. The outer loop continues indefinitely, responding to queries from registered clients.

Require: C^q, $\mathcal{X}$, s, ϵ
1: **loop**
2: $Y \leftarrow$`await query(`$C_j \in C^q$`)`
3: **if** $\neg$`verify(`C_j`)` **then**
4: **return** $\emptyset$
5: **end if**
6: $\mathcal{X}, \mathcal{M} \leftarrow$`local(`$Y$`)`
7: $W \leftarrow \emptyset$; $\sigma \leftarrow 0$; $r \leftarrow 0$
8: **for** $\bar{w}_i \in \mathcal{X}$ **do**
9: $W \leftarrow W \cup \{w_i\}$
10: $\sigma \leftarrow \sigma + w_i$
11: **end for**
12: $\bar{w} \leftarrow (1/|W|) \sum_i w_i$
13: **for** $w_i \in W \setminus \{\max(W)\}$ **do**
14: **if** $|w_i - \bar{w}| < \epsilon$ **then**
15: $\sigma \leftarrow \sigma - w_i$; $w_i \leftarrow 0$
16: **end if**
17: **end for**
18: **for** $w_i \in W$ **do**
19: $r \leftarrow r + (w_i/\sigma) M_i(Y)$
20: **end for**
21: **return** r {Does not end the loop.}
22: **end loop**

Algorithm 2 Local client response updates.

Require: C^u (participating clients), Y (submitted query)
1: **loop**
2: Initialize the list $H \leftarrow \emptyset$.
3: **for** $C_i \in C^u$ **do**
4: $M_i(Y), \bar{w}_i \leftarrow$`async update(`$C_i, Y$`)`
5: $\mathcal{M} \leftarrow \mathcal{M} \cup \{M_i\}$
6: $\mathcal{X} \leftarrow \mathcal{X} \cup \{\bar{x}_i\}$
7: **end for**
8: **return** $\mathcal{X}, \mathcal{M}$ {Does not end the loop.}
9: **end loop**

C. Security and Privacy Analysis

Security Analysis. Federated learning models are subject to fabrication attacks from rogue clients. That is, a rogue client can provide inaccurate model parameters or fabricate the centroid point to mislead the federated model. Rogue clients constitute a limitation of our model, which minimizes data traffic workload and privacy loss from clients. The presented federated interpolation scheme follows a single-update approach with multiple queries. The server must be equipped with a solid authentication system for clients, encrypting the data exchange between the server and a client and securing clients from remote network attacks. While important, addressing these three requirements are beyond the scope of this work.

Detecting fabricated model parameters can be done through a different federated learning approach in which the server manages the learning process. The server should dispatch learning tasks to clients. Detecting rogue clients can be challenging in this setting. Further, we avoided this approach because of the additional data communication cost.

Privacy Analysis. Clients are only required to interact with the trustworthy but curious server. No inter-client interaction is required. We define the trade-off between privacy and the utility of a federated learning system $\mathcal{S}$ for each client as

$$\mathcal{L}_p(D_i, f_{C_i}(x)) \propto \frac{1}{\mathcal{L}_u(f_{C_i}(x))}, \tag{2}$$

where the local data model $f_{C_i}(x)$ may differ from the communicated model $M_i(x)$, $\mathcal{L}_p$ is the privacy loss, and $\mathcal{L}_u$ is the utility loss. The utility loss is measured in terms of an error metric. The privacy loss is measured as the proportion of a data set D_i exposed to any other client or the server:

$$\mathcal{L}_p(D_i, f_{C_i}(x)) = \frac{|D_i^k|}{|D_i|}, \tag{3}$$

where D_i^k is the exposed portion of D_i. Hence we focus on minimizing the number of data points shared by clients that produce data.

In the federated interpolation model, no single raw data point is exposed. The privacy loss is 0.

V. Experimental Results

This section presents experiments to evaluate the feasibility of the federated model interpolation scheme in Section IV. We are primarily interested in knowing whether federated model interpolation is capable of maintaining a high prediction accuracy, ideally as good as the one achieved if the entire dataset was available for model generation.

To investigate the feasibility, we test our scheme against an extreme case in which a single client owns all the dataset and can train and test a model on the available data. In contrast, federated model interpolation will not have access to the data. We simulate the architecture of our federated learning system (Figure 1), which builds the federated interpolant and sends queries to the clients based on the proximity of the query to the client centroids.

A. Evaluation setup

Datasets and the Learning Task. Two real-world tabular datasets are used to evaluate the accuracy of classification using federated interpolation. First, we use a small COVID-19 metadata dataset produced for X-ray images collected from patients with suspected respiratory disease [12]. The learning task is to to classify patients as surviving or not. Table I shows the selected features from the dataset. These features were selected such that all rows have data points for the selected features.

Second, we use the large KDD CUP 99 dataset [13], [18] for intrusion detection. The KDD CUP 99 dataset has 40 features and a single target variable. The features (as described in [18]) capture parameters of network activity by various network services (such as FTP, SMTP, and HTTP). The learning task is a multiple target classification of a network request into benign or a specific type of network intrusion attack (such as buffer overflow).

Simulation Environment. To form the updating clients (that supply data), we divide the dataset into k clients. For the COVID-19 dataset, k is the number of countries in the dataset, specifying each client as a separate country. For the KDD CUP 99 dataset, the clients are selected based on the attack type. For each client we generate a model of the data to perform the classification task. The machine learning algorithm is an implementation of support vector classifier with a linear kernel [19].

Measuring Accuracy. We compare federated learning with both datasets against a single data owner model. In a single-owner scenario, one entity can directly create a model from all data points and answer client inquiries. In contrast, federated learning aims to derive insights from multiple clients without directly accessing their datasets, yet it aims to achieve accuracy comparable to the single-owner model. We investigate whether federated model interpolation can produce meaningful results. If so, what is the accuracy level achieved compared to the single-owner model?

Sex	Age	COVID19	PCR+	ICU	Tube	Temp.	Survival
1	70	1	1	1	1	40	0
0	70	0	1	1	1	38	0
1	58	1	1	1	1	40	1
0	30	1	1	0	0	39	0
1	45	0	1	1	1	38.5	0
1	46	1	0	0	0	39	0

TABLE I
A sample of rows in the COVID-19 metadata dataset developed by the University of Montreal. 434 data points are used to evaluate the federated interpolation scheme.

B. Evaluation Using COVID19 Dataset

Nonoverlapping Clients. This section presents the accuracy of the federated interpolation scheme to learn an aggregated model from local clients with at least a single *nonoverlapping* feature. That is, for each client, there exist a feature X_{ij} (from the n available features) such that $X_{ij} \neq X_{kj}$ for all $k \neq i$. The results are compared with the accuracy of individual local client models and a single global model trained on the union of all client datasets. For our experiments, we divided the client datasets according to the age feature as specified in Table II.

Client	Age Interval	Points
1	[0, 50)	136
2	[50, 60)	90
3	[60, 70)	81
4	[70, 110)	127

TABLE II
Synthetic clients created based on the COVID19 metadata dataset. Clients are chosen to balance the number of data points.

The dataset reflects real-world cases that the University of Montreal collected. However, we divided the dataset based on age intervals to create clients and simulate building a federated model. The choice of age intervals is to balance the number of data points from each client. Further, since age is a decisive factor in COVID19 survival, we selected the age feature as a distinctive factor to create disjoint client datasets.

Evaluation Approach.

The learning task is to fit a model that predicts the survival of a patient. For local model generation, a combination of linear regression, decision trees, and support vector classifiers

(SVC) with a linear kernel [20] were used to fit the dataset and produce predictions of survival. Depending on the outcome of the algorithm, the best accuracy was chosen. Initially, the entire dataset was made available to a single global server where the data fitting using SVC was done. Next, the datasets were split according to Table II and each client dataset was fit using the support vector classifier algorithm. Finally, a federated learning server was used to fit a federated model, interpolating client models based on the scheme defined in Section IV. Accuracy of the model predictions $M(x_i)$ in all experiments were measured using accuracy (ratio of correct predictions to all predictions).

$$\text{Accuracy} = \frac{1}{N}\sum_{i=1}^{N}\delta\big(M(x_i) = f(x_i)\big),$$

where N is the number of data points in the dataset, and $\delta(E) = 1/0$ if the expression E is true/false. In each experiment, 30% of the dataset is randomly chosen as a test set. Further, 7-fold cross-validation is used in which the test and training sets are refreshed to avoid fitting to a specific subset. When fitting a federated model, the test dataset is formed by computing the union of all local test sets.

Table III shows the accuracy of model predictions in the three settings (global server, local models, and interpolation server).

Server	Accuracy	Points Used	Models Used
Global	0.750	434	1
Client 1	0.756	136	1
Client 2	0.667	90	1
Client 3	0.667	81	1
Client 4	0.714	127	1
Federated (Average)	0.693	4	4

TABLE III
ACCURACY RESULTS FOR THREE BINARY TRAINING SETTINGS (GLOBAL SERVER, LOCAL MODELS, AND INTERPOLATION SERVER) FOR THE COVID19 METADATA DATASET.

Overlapping Clients. With overlapping clients, four distinct clients were created by randomly sampling the entire dataset. Two additional clients were created by mixing data points from the first four, each receiving two random halves from two other clients. In these experiments, no significant change in the values of Table III were observed.

C. Evaluation Using KDD CUP 99

The KDD CUP 99 dataset is an intrusion detection dataset. For our experiments, we used 40 features from the dataset with a total of 494021 data points.

The clients for this dataset were real clients that had reported data points independently. Although the original dataset was not intended for a federated learning setting, the dataset was produced in a distributed environment with multiple service reporting network activities. Thus we isolated the client datasets based on the service feature in the dataset. For example, network activity reported on the `http` service comprises a separate client dataset. Using data points reported on specific services, participating clients are nonoverlapping.

The learning task is to fit a decision tree to classify the type of network activity. There are 23 possible values for the type of network activity, such as normal, multihop, Nmap, and rootkit. Each class is assigned an integer value in the range 0–23. No particular order is used in assigning the values. The results of this experiment are in Table IV showing the accuracy metric.

Sensitivity to Size of Dataset. To test the sensitivity of the federated interpolation scheme to the size of the dataset, we repeated the experiment with the KDD CUP 99 dataset in three settings. A minimum number of data points were required for a client to participate in the learning process in each setting. Experiments 1, 2, 3, 4, and 5 require at least 20, 100, 500, 1000, and 5000 data points, respectively. In each setting, we trained a decision tree. Figure 2 shows that the average accuracy of the local model is mildly improved when clients with larger datasets participate. The increase in the number of client data points required contributed to an 8% increase in the accuracy of the federated interpolation scheme, as observed in Experiments 3, 4, and 5. However, Experiments 1 and 2 show that the federated interpolation scheme has the opportunity to improve accuracy when a large number of clients participate, even when each client does not necessarily provide a large dataset. Despite the average local model accuracy declines, the federated interpolation relatively increases in accuracy with many clients. The slight accuracy decrease in Experiment 2 compared to Experiment 1 shows that the higher number of clients improves the accuracy. The decrease could be due to the improved diversity that the more significant number of local models represents.

Server	Accuracy	Points Used	Models Used
Global	0.999	494021	1
Local 1	0.992	110893	1
Local 2	1.000	281400	1
Local 3	0.998	1642	1
Local 4	1.000	7237	1
Local 5	0.999	4721	1
Local 6	0.999	9723	1
Local 7	1.000	5863	1
Local 8	1.000	64293	1
Local (Average)	0.998	494021	8
Federated (Average)	0.909	8	8

TABLE IV
ACCURACY RESULTS FOR THREE MULTICLASS TRAINING SETTINGS (GLOBAL SERVER, LOCAL MODELS, AND INTERPOLATION SERVER) FOR THE KDD CUP 99 DATASET WITH EIGHT CLIENTS. THE AVERAGE VALUES INDICATE AVERAGE SCORE ACHIEVED USING TEST SETS FROM THE CLIENTS. FEDERATED LEARNING ACCURACY IS 9% LOWER ON AVERAGE, RELATIVE TO A SINGLE GLOBAL MODEL.

VI. CONCLUSION AND FUTURE WORK

This work presented a new approach to aggregating multiple heterogeneous local models received from independent clients, aiming to produce a global model. The global model is maintained by a server that is assumed to be trustworthy, which may also violate the privacy of data collected from

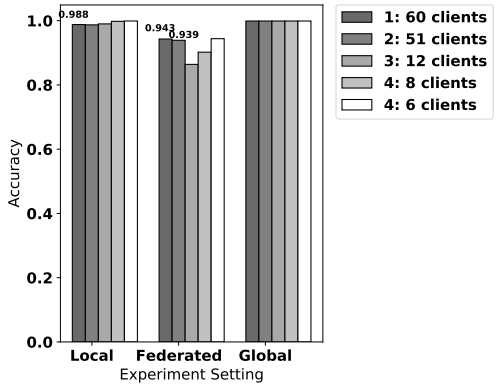

Fig. 2. A comparison of classification accuracy in two sets of experiments. Experiments differ in the number of data points that a client must have to participate in the learning process. Clients must have at least 20, 100, 500, 1000, and 5000 data points to participate in Experiments 1, 2, 3, 4, and 5, respectively.

clients. The results show that the server does not need any data points or the model parameters to be available at the time of processing requests. Instead clients can provide computed values according to each query.

For future work we envision developing the model towards a hierarchical learning system. The single central server can be replaced with multiple servers, each responsible for models of a subset of clients. The system can respond to queries according to a specific server or with a combined global metamodel of all models produced by individual servers. This system model can incorporate the remedies mentioned earlier, such as multiple centroids and models from a client, and fuzzy classification of a query Y when centroid distances are not sufficiently discriminating.

REFERENCES

[1] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, pages 1273–1282. PMLR, 2017.

[2] Bargav Jayaraman, Lingxiao Wang, David Evans, and Quanquan Gu. Distributed learning without distress: Privacy-preserving empirical risk minimization. *Advances in Neural Information Processing Systems*, 31, 2018.

[3] Yuheng Zhang, Ruoxi Jia, Hengzhi Pei, Wenxiao Wang, Bo Li, and Dawn Song. The secret revealer: Generative model-inversion attacks against deep neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 253–261, 2020.

[4] Jenny Hamer, Mehryar Mohri, and Ananda Theertha Suresh. Fedboost: A communication-efficient algorithm for federated learning. In *International Conference on Machine Learning*, pages 3973–3983. PMLR, 2020.

[5] Sebastian Caldas, Sai Meher Karthik Duddu, Peter Wu, Tian Li, Jakub Konečnỳ, H Brendan McMahan, Virginia Smith, and Ameet Talwalkar. Leaf: A benchmark for federated settings. *arXiv preprint arXiv:1812.01097*, 2018.

[6] Bjarne Pfitzner, Nico Steckhan, and Bert Arnrich. Federated learning in a medical context: A systematic literature review. *ACM Trans. Internet Technol.*, 21(2), jun 2021.

[7] Manjula A. Iyer, Layne T. Watson, and Michael W. Berry. SHEPPACK: A Fortran 95 Package for Interpolation Using the Modified Shepard Algorithm. In *Proceedings of the 44th Annual Southeast Regional Conference*, ACM-SE 44, page 476481, New York, NY, USA, 2006. Association for Computing Machinery.

[8] William I. Thacker, Jingwei Zhang, Layne T. Watson, Jeffrey B. Birch, Manjula A. Iyer, and Michael W. Berry. Algorithm 905: Sheppack: Modified shepard algorithm for interpolation of scattered multivariate data. *ACM Trans. Math. Softw.*, 37(3), September 2010.

[9] Yishay Mansour, Mehryar Mohri, Jae Ro, and Ananda Theertha Suresh. Three approaches for personalization with applications to federated learning, 2020.

[10] Mikhail Yurochkin, Mayank Agarwal, Soumya Ghosh, Kristjan Greenewald, and Nghia Hoang. Statistical model aggregation via parameter matching. *Advances in Neural Information Processing Systems*, 32:10956–10966, 2019.

[11] Zonghao Huang, Rui Hu, Yuanxiong Guo, Eric Chan-Tin, and Yanmin Gong. Dp-admm: Admm-based distributed learning with differential privacy. *IEEE Transactions on Information Forensics and Security*, 15:1002–1012, 2019.

[12] Joseph Paul Cohen, Paul Morrison, Lan Dao, Karsten Roth, Tim Q Duong, and Marzyeh Ghassemi. COVID-19 image data collection: Prospective predictions are the future. *arXiv 2006.11988*, 2020.

[13] Stephen D. Bay, Dennis Kibler, Michael J. Pazzani, and Padhraic Smyth. The UCI KDD archive of large data sets for data mining research and experimentation. *SIGKDD Explor. Newsl.*, 2(2):8185, December 2000.

[14] Wei Liu, Li Chen, Yunfei Chen, and Wenyi Zhang. Accelerating federated learning via momentum gradient descent. *IEEE Transactions on Parallel and Distributed Systems*, 31(8):1754–1766, 2020.

[15] Zhaoxian Wu, Qing Ling, Tianyi Chen, and Georgios B Giannakis. Federated variance-reduced stochastic gradient descent with robustness to byzantine attacks. *IEEE Transactions on Signal Processing*, 68:4583–4596, 2020.

[16] Yue Tan, Guodong Long, Lu Liu, Tianyi Zhou, and Jing Jiang. Fedproto: Federated prototype learning over heterogeneous devices. *CoRR*, abs/2105.00243, 2021.

[17] Sebastian Claici, Mikhail Yurochkin, Soumya Ghosh, and Justin Solomon. Model Fusion with Kullback-Leibler Divergence. In *International Conference on Machine Learning*, pages 2038–2047. PMLR, 2020.

[18] Mahbod Tavallaee, Ebrahim Bagheri, Wei Lu, and Ali A. Ghorbani. A detailed analysis of the KDD CUP 99 data set. In *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, pages 1–6, 2009.

[19] Chih-Chung Chang and Chih-Jen Lin. Libsvm: a library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)*, 2(3):1–27, 2011.

[20] Chih-Chung Chang and Chih-Jen Lin. Libsvm: A library for support vector machines. *ACM Trans. Intell. Syst. Technol.*, 2(3), May 2011.