Federated AI for the Enterprise: A Web Services based Implementation

Dinesh Verma
IBM T. J. Watson Research Center
Yorktown Heights, NY 10598, USA
Email: dverma@us.ibm.com

Graham White
IBM Research UK
Hursley Park, UK
Email: gwhite@uk.ibm.com

Geeth de Mel
IBM Research UK
Hartree Center,Warrington, UK
Email: geeth.demel@uk.ibm.com

***Abstract*— Many enterprise solutions can be driven off Machine Learning models that are created from the data belonging to an enterprise. However, many enterprises can not share data freely across different locations due to regulatory restrictions, performance issues in moving large data volumes, or requirements to maintain autonomy. In these cases, the enterprise can benefit from the concept of federated learning, in which machine learning based models are created at multiple different sites, and then combined together at a federation server without the need to share data. In this paper, we describe a web-services based implementation of the federated learning system, focusing on the problems enterprises encounter in used of distributed data, and discussing how we solved those problems.**

I. Introduction

The rapid advancement in the technology during the last decade has enabled enterprise applications to use Artificial Intelligence (AI)—specifically Machine Learning (ML)—for real-time decision making; in such applications, the operations depend on creating an AI model which is built during a learning process using training data. The AI model is then used in the decision making process to perform inferences about suitable actions to take—especially in previously unencountered situations. These AI models are typically used as part of a larger pipeline consisting of a mix of other AI models and traditional programming paradigms. Thus, the effectiveness of the AI based operations depends primarily on the quality and availability of training data.

In many enterprise contexts, training data can be obtained and stored centrally, e.g., in a data lake. It is straightforward to build an AI model when the training data is stored in a central location using a variety of existing algorithms designed for scalability and performance. However, there are many other cases in an enterprise where training data is distributed across many different locations. The reason for training data to be split can be varied (e.g., mergers and acquisitions bringing data distribution to an enterprise) and there are instances where training data can not be moved to a central location (e.g., regulated industries may not be able to move data across geographical boundaries).

*This research was sponsored by the U.S. Army Research Laboratory and the U.K. Ministry of Defence under Agreement Number W911NF-16-3-0001. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Army Research Laboratory, the U.S. Government, the U.K. Ministry of Defence or the U.K. Government. The U.S. and U.K. Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

To deal with such situations, algorithms have been developed that train models at many different locations and then fuse them together to generate an overall model. We call this process *Federated AI*, and it can be used in many contexts: *(a)* multi-domain collaborative operations at the tactical edge [1], [2]; *(b)* blockchain enabled applications [3]; and *(c)* multi-agencies cohesion [4] to name a few. In addition to the algorithmic challenges involved in merging models, a Federated AI implementation requires addressing many other challenges that arise in a practical implementation, and those issues are addressed in this paper.

Web Services provide a convenient mechanism for implementation of federated learning in the enterprise context. This is because, a web service based architecture provides service mix-and-match capability to federated learning so the learner can seamlessly utilize human-invoked functions along with automated service invocations. Additionally, web service frameworks enable the traversal of enterprise firewalls using a protocol that is usually known, trusted and enabled across the enterprise boundaries, thus making the distributed learning process secure.

We would like to distinguish Enterprise Federated AI from related but distinct area of mobile device oriented federated learning that deals with the situation of massively distributed data, e.g. learning is performed on many mobile phones [5], [6]. While there is some commonality in the basic approach for combining models, the practical challenges that need to be addressed for these two environments are very different. Federated learning for mobile devices requires creating models for large-scale distributed data that would usually be generated by the same application, leading to homogeneous data but very small amounts of data at each device. In the context of enterprises, the data is collected independently at each site resulting in a few silos of data, each containing large amounts of data, but data at each site may be heterogeneous in content and semantics.

The rest of the document is organized as follows: In Section II we discuss key challenges in enterprise federated learning. In Section III we highlight common scenarios where the enterprise needs federated learning, and we then argue for a client-server based approach for solving the problem in Section IV. We discuss specific algorithms that are needed to realize the service based federated learning in Section V. In Section VI, we conclude the document by providing final remarks and areas for future exploration.

TABLE I

DIFFERENT TYPES OF TRAINING DATA OPTIONS

Category	Input	Output
I	Featured	Known
II	Raw	Known
III	Featured	Unknown
IV	Raw	Unknown

II. CHALLENGES IN ENTERPRISE FEDERATED LEARNING

The model for federated learning we envision is shown in Figure 1. Each site has some part of the overall training data available locally. Each site also has a local web-service that provides an interface to an application that can access local data, train a model, and allow local users and trusted remote parties to invoke selected functions at the local service. One of these sites also acts as a Fusion Service site. The Fusion Service site allows other sites to access its Fusion Services. Furthermore, we assume the other sites will allow the Fusion Service to access its site. Since these are enterprise environments, each of the sites is protected by a firewall that only allows limited access to trusted parties and services.

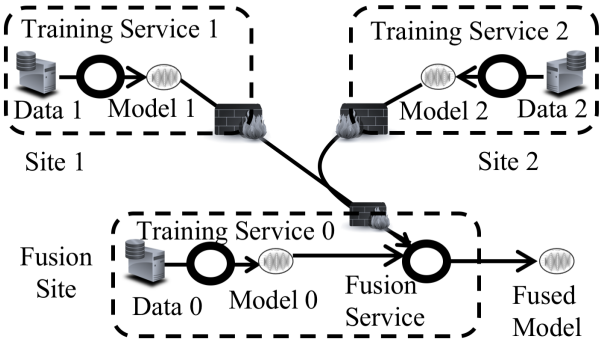

Fig. 1. The model for Fusion AI. Training data is available at many different sites, and a training service is used to create model locally. A fusion service at one of the sites helps coordinate the fusion of the models.

Any machine learning problem can be viewed as the task of mapping an input to an output. For different types of machine learning applications, training data can be divided into one of four categories depending on the characteristics of the input and output in the training data set as shown in Table I. Some of the common machine learning tasks and the category in which they would fall are shown in Table III.

An input for an AI/ML problem may be raw or consist of features. A featured input can be realized as a tabular data where each data point consists of a set of features i.e. a row in the table. The input is raw if its does not consist of features, e.g. it may consist of images, sounds, or text. In many cases, these raw inputs are mapped to features. In other cases they may be put into an AI/ML model without trying to extract the features that get extracted as part of the learning process. For some types of problems, the training data may explicitly identify an output, e.g. a label for classification or an indication whether the data point is anomalous or not. In other cases, the training data may not consist of an output. The latter are used for clustering the inputs and finding anomalous instances.

TABLE II

CATEGORIES OF SOME COMMON MACHINE LEARNING TASKS

Task	Category	Explanation
Credit Risk Analysis	I	Input is tabular with features as columns and output is a value
Linear Regression	I	Input has features, output is predicate
Image Classification	II	Input is raw images without well-defined features and output is a class
Speech to Text	II	Input is audio of speech output is text symbols
Fraudulent Purchase Detection	III	input is tabular each column is a feature output is unusual activity
Document Clustering	IV	Input is unstructured text output is category of document
Engine Malfunction Detection using Sounds	IV	input is audio output is binary indicating anomalous sounds

In this model, many different challenges occur for federated learning environments for each of the different categories of training data. Some of these challenges are described below.

Varying Data Quality: For training data that consists of raw input (categories II and IV in Table I), the modalities could significantly vary from site to site. This will have an impact on the global model the distributed learner creates. For example, let us assume a scenario where the purpose of the learner is to create an acoustic model about mechanical failures of equipment using observations stored in two locations. Let us also assume that the acoustic data is captured as WAV and MP3 at each site respectively. Given that fact that WAV files preserve all data whilst MP3 files are lossy, the individual models generated out of these two sources could vary significantly. Thus, strategies need to be considered for how best to merge these models. Similar observations could be made about other data quality issues—e.g., resolution differences in visual data resulting in varying weights for similar features.

Linguistic and Lexicon Mismatch: For training data that consists of known output (categories I and II in Table I), the labels that are assigned to the outputs at different sites may differ; similarly, training data that consists of featured input (categories I and III in Table I), the names of the features that are used at different sites could also be different. This is due to the varying terminologies used at different locations, thus resulting in ambiguous and conflicting labeling conventions.

Incomplete Knowledge: Data at each site may only contain a fragment of the whole data landscape. Thus, there could be a missing *feature problem*, i.e., for training data that consists of featured input (categories I and III in Table I), some features may only be present at some of the sites.

Moreover, for training data that consists of known output (categories I and II in Table I), some of the labels may be missing at some of the sites. We refer to this as *missing classes* problem and this can have a significant impact on the accuracy of federated learning [4]. Finally, there may be situations in which some *values are missing* from the underlying schema, e.g., training data that consists of featured input (categories I and III in Table I), if missing values are found across sites it must be ensured that these are resolved in the same way at each site.

Skewed Classes: For training data that consists of known output (categories I and II in Table I), the distribution of classes in some sites may be entirely different from the distribution of the classes in other sites. This results in each site having a different mixture of classes in the training data.

Skewed Features: For training data that consists of featured input (categories I and III in Table I), the data at different sites may be collected for different regions of the feature space. Extrapolating across different regions of applicability can cause mistakes in the estimation of the AI model [7].

Size Differences: Different sites may have collected different volumes of training data, which may make some trained models be better than others.

Synchronization: Depending on the level of coordination among different sites, they may be able to conduct their training algorithms in synchronized manner with each other. In other cases, such synchronization may not be possible and model training may need to happen in an asynchronous manner. Asynchronous training may particularly be required in cases where one site has more/less processing power or capability than another site to avoid one partner unduly having to wait for the other to complete each round of training.

Service Access: Some sites may be located within the same access domain, and the fusion service can invoke commands on the training service. In other cases, the access of the fusion service to the training service may be limited.

Exchange Trust: While the sites may not be able to share the complete data with each other, in some cases the sites may be allowed to share small pieces of data with each other. Such an exchange of data can significantly improve the process of model building [4]. In other cases, exchange of raw data may not be permitted but a synthetic representation of the data, generated using techniques like a generative adversarial networks (GAN) [8] may be permitted. The trust differences among the sites needs to be taken into account.

In any scenario where Fusion AI is being used, these differences need to be accounted for.

III. Scenarios for Fusion AI

Within an enterprise context, the need for Fusion AI arises in many different situations. In this section, we look at some of those scenarios, and discuss the unique requirements for federated learning that arises in those situations.

In an ideal world, where network bandwidth is plentiful and cheap, and there are no restrictions on moving data across different sites, the ideal way to build AI models would be to migrate all the data over to a single site, and to train the model there. However, there are many situations in which such a process is not feasible.

A. Subsidiaries and Franchises

Many enterprises are arranged as loose conglomerations of subsidiary companies. In many multi-national corporations, operations in different countries may be structured as different subsidiaries. While there is a loose coordination among the different subsidiaries, they are often operated in an independent manner. In some cases, the subsidiaries may be subject to different regulatory restrictions, and export of data beyond some administrative boundaries may be prohibited. Multi-national banks and multi-national corporations often fall into this category.

In a subsidiary model, some of the IT functions may be controlled centrally, and in some environments, the schema can be assumed to be the same for all of the different sites. However, the volume of data that may be collected for each of the different sites may be very different. Similarly, the characteristics of data available at different sites may vary significantly, e.g. one subsidiary may have mostly young professional as clients while the other subsidiary may have mostly older retirees as clients.

The operation of many banks, governments and multi-national companies can be viewed as instances of subsidiaries. Subsidiaries would typically trust each other, e.g. allow access to each other services by opening ports within their services.

The operation of many other types of large businesses can be viewed as instances of franchises. There may be a central model available from the franchise owner with franchisees able to extend this using their own data. Groups of franchises may also choose to work together but in many situations franchisees will want to protect their business data by keeping it private to themselves or their local group of franchisees.

B. Mergers and Acquisitions

There is a significant number of mergers and acquisition activities which occur among enterprises, specially with big or medium enterprises acquiring smaller enterprises. Almost a thousand mergers and acquisitions happen annually [9], with a merger bringing two roughly equal sizes enterprises into a single unit, and an acquisition bringing a smaller enterprise within a bigger enterprise. When the company being acquired is of a reasonable size, it is complex to consolidate the different IT systems of different component companies together. While the multiple IT systems are converged into a single IT component eventually, such convergence can frequently take years.

While integration is underway, the schema and data collected by different components of the company could be very different. In these cases, the schema of the data collected

at different sites may be different, in addition to the other challenges faced by an enterprise consisting of multiple subsidiaries.

C. Outsourced Operations

A different case of split data is encountered by a company which provides outsourced services to its enterprise clients. As an example, a services company may be running the IT operations of several banks. The applications that are supported in many of these companies are the same, and insights used across support of different applications be used to improve operations across other organizations. While the banks may allow the sharing of insights and models learned from their operational data to have the outsourced company improve its operations, they are unlikely to allow the transfer of raw data beyond their premises.

The challenges in this environment is that the data collected by different companies may be in different formats, the banks may only allow models to be moved about in an asynchronous manner after they are fully trained, and that the fusion service of the outsourced company may need to work without any validation data of its own. The interface between the Training Service and Fusion Service may also be constrained, with the Training Service only be allowed to make outbound requests to the Fusion Service.

D. Operations at Scale

In some organizations, operations happen at such a massive scale that operational data needs to be stored independently at multiple locations. The volume of data does not permit it to be moved easily across different sites. In any large telecommunications company, a significant amount of network management and network operations data is collected at multiple regional centers. This data is very massive to be moved across to a central location. In other cases, when the number of subscribers is large, information about calls is stored at multiple different locations. With billions of call records being generated every day, the call records are stored in many different locations, and only aggregated statistics moved around to a central location. In those cases, any AI model built across different locations need to move analytic logic around, instead of moving the data around. The same holds true for AI model.

The at-scale operation provides the most congenial environment for federated learning. The data is usually in a consistent format with the same schema, the training process could be synchronized, and the services can invoke each other.

E. Consortium

A consortium sees many different companies come together to share information or details in a specific field. As an example, a consortia of banks may be interested in sharing insights so as to better provide fraud detection. They may be able to share and provide models to each other, but not be able to share the entire amount of data in clear. In other cases, the consortium may be a group of medical companies, which are able to share data with each other, but only if it is anonymized and aggregated. They would like to work with each other while preserving the privacy of their individual members.

Members in a consortium may not completely trust each other, and may only be able to share limited amounts of information with each other.

F. Regulated Industries

In some industries, regulations prevent migration of raw data from a location that is collecting data to the location that can analyze it to train a model. A typical case would be large hospital systems which see patients as well as conduct medical research. Usually, the research organization would be different than the clinical (out-patient) organization. Data collected at different clinical groups may be stored separately depending on the type of access controls required by the system. An AI model in these cases would be run by the research organization. Yet the research organization would not be allowed to access the raw data.

One way to address these requirements would be to transform the data to hide any personal information before creating an AI model from it. However, implementing a variation of Fusion AI, which would be able to process raw data at each of the clinical site, is more likely to provide a more accurate model.

G. Military Coalitions and Multi Domain Operations

A military coalition can be viewed as a special case of a consortium, where the participants are armed forces of different countries, as opposed to a commercial enterprise. However, military coalitions face another unique challenge. All members of a typical commercial consortium would have a presence on the Internet, and different members will be connected together via a reasonably high speed connection (typical Internet speeds). However, military coalitions frequently operate in remote areas, where connectivity is provided by satellites, or using ad-hoc wireless networks, which offer very limited bandwidths, high latency and very low reliability. With increasing computational capability available at the different vehicles, hand-held electronics, and other equipment used by the military, bandwidth considerations move them towards the concept of federated learning.

A more comprehensive overview of the challenges involved in enabling AI enabled systems at tactical edge for coalitions is provided in [10]. The challenges are those of a consortium operating in a low bandwidth environment.

A multi-domain operation consists of different domains of the military belonging to the same nation which are working together. A domain in this case refers to one of space, air-force, naval forces and land-forces. Multi domain operations share many challenges with coalition operations, with the additional complexity of different types of assets, and differences in the uncertainty of information that results from disparate assets.

H. A Common Model

For each of the scenarios that we have described above, we can assume that the system used for Fusion AI consists of a set of training services and an instance of a Fusion Service. All the training services and fusion service will be implemented as web-service, with well-defined interfaces for invoking the functions on them. The physical architecture of the canonical model for any scenario would be as shown in Fig. 2

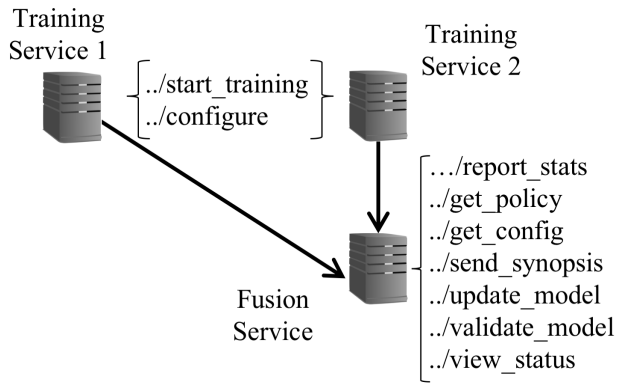

Fig. 2. The canonical model for Fusion AI. A set of Training Services interact with a Fusion Service in order to train model without moving data.

The training service provides only two service calls, one to configure it, and the other start the training process at each of the clients. These commands can be invoked manually, or in some cases by the fusion server. The fusion server provides a richer set of commands which are invoked by the different training services. The manner in which these services are invoked, and how the different functions are called in sequence is described in the next section.

IV. Fusion and Training Web Services

In order to provide a good practical solution that can be used to address the large number of scenarios and challenges in Fusion AI, we assume the training service and fusion service go through a sequence of processing steps as shown in Fig. 3 (validation step not shown).

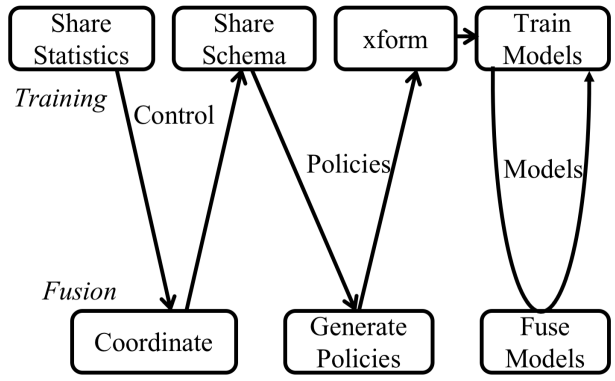

Fig. 3. The sequence of processing steps for Training and Fusion Service.

TABLE III
Description of Training Data

Category	Input Description	Output Description
I	Feature names	Label Values
II	Type	Label Values
III	Feature names	NA
IV	Type	NA

When the training service is invoked via the configuration command, it analyzes the statistics of the locally available training data. It collects the description of the training data that it has locally, that consists of the information as shown in Table III.

Having obtained the statistics from each training service, the fusion server will create a set of control and configuration parameters for the fusion AI training session. This training session includes instructions for obtaining training policies, instructions for facilitating a data exchange, if needed, as well as information about the fusion algorithm and the parameters needed for the fusion process. The fusion service sends this control and configuration information to the requesting training services.

On the receipt of the control information, each training service contacts the fusion service to get a set of transformation policies. The fusion server compares the data it has locally with the provided statistics, and uses it to generate a set of policies for transformation of data at the training service. The goal of these generated policies is to get data from each of the different training services into a common schema. The policies that will be sent to the training service will include instructions for changing the type of the raw data (e.g. convert .jpg images into .png, or convert .avi sound files into .wav files etc.), relabeling the features to a common set of names, and relabeling the output label values into a different common set. The algorithms for generating these policies are described in more detail in [11].

Upon receipt of the policies, the training service uses the policies to convert the local training data into a specific format. The previously received control information instructs the training server about the operations it should conduct before starting the fusion, e.g. in some types of fusion processes it may need to send a small sample of its data set or a generator for representative synthetic data. The control information may also contain information about batch-sizes and number of iterations the training service may need in order to conduct a successful model training exercise. With the receipt of the control parameters, the training service goes through the training stage, working with the fusion service in the fusion stage. Once the training has been completed, the fusion server may validate the model and compute its performance metrics. This may also include synchronizing with different training services.

The invocation of different interfaces exposed by the fusion service once the training service is invoked at any location is as shown in Figure 4. The task performed by each of the services is discussed in Section IV-A

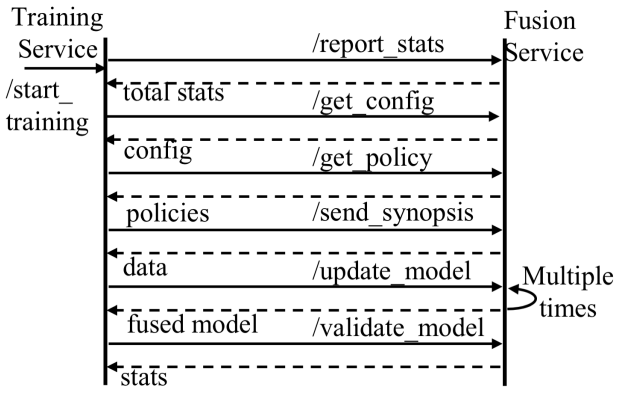

Fig. 4. The invocation of fusion services once training service is called.

A. Service APIs

The training service exposes two interfaces, one to initiate the training process (`/start_training`), and the other to manage the configuration of the service at the local site (`/configure`). The configuration of the training site comprises of the location of the training data and the training service. The configuration interface allows both the viewing and the update of the configuration of the service. When the training process is initiated, the sequence of steps described in Fig. 4 is initiated.

The Fusion Service exposes a rich interface consisting of several services needed to address the different aspects of the challenges encountered in federated learning. The interfaces exposed by the fusion service include:

`report_stats`: used by a training service to provide the aggregate statistics of its local data. This includes a description of its schema, the type of training data it has, and if labeled the number of data points for each output label. The statistics are used by the fusion server to determine the best way to run a fusion algorithm so that it works well with data that is partitioned differently.

`get_config`: called to get the control configuration parameters for a fusion AI machine learning session. The control and configuration parameters are based on the attributes and statistics of all participants in the training site. The configuration include instructions on how to implement the policy generation, data synopsis exchange and fusion algorithm.

`get_policy`: called to get the transformation policies from the Fusion Server. The Fusion Server controls how the data from different training sites ought to be transformed so that it can correspond to a common schema and format for input training data. The policy may also indicate an approach to change the labels to get the canonical data for training.

`send_synopsis`: called optionally to get a brief portion of data from each training site and share it with other sites. This allows the sites to work around problems of missing classes or skewed data. The mechanisms for exchange, whether using some kind of universal summarization like a core set for machine learning [12], a generator or a small sample of the data, will be determined during the policy separation phase.

`update_model`: used during the training phase of the federated learning. At each invocation of this interface, the fusion server receives the results of local training at each site, and returns the fused model that comes from integrating the other training sites.

`validate_model`: is used after the training phase to validate the model, and to check its accuracy. For the validation phase, a distributed validation can be done to check the performance of the model on the validation data at each location.

`view_status`: provides a management interface to check on the status of the learning session and its progress.

B. Software Structure of Services

There are many AI models and many types of conditions under which Fusion AI has to operate. Given the wide range of scenarios for Fusion AI and the various combinations of different challenges that need to addressed. In order to deal with those challenges, the software structure shown in Fig. 5 is used. The software structure is built behind a web-services framework such as Django [13], Express or Spring.

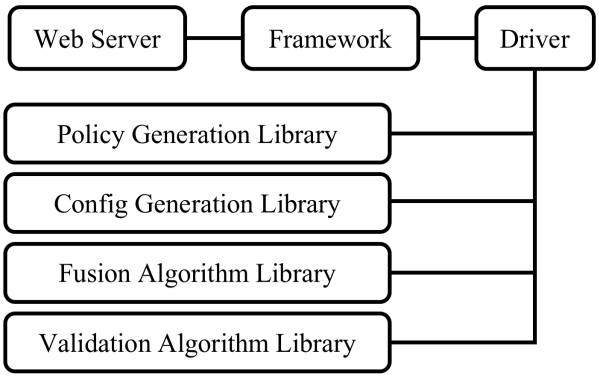

Fig. 5. The structure of the software.

The software library can be common across the training service and the fusion service, and implements a suite of functions corresponding to the phases described in Fig. 3. The driver is different depending on the client or the server and calls different functions in the libraries depending on its location.

Each of the library components shown on the right-most side of Fig. 5 consists of the same type of algorithm, divided into six routines, four belonging to the training service and two belonging to the fusion service functions. The three

client side components implement a function that is called by the client (i) during the initial invocation of the client (ii) when the client needs to iterate through a step in the phase shown in Fig. 3 before sending a message to the server and (iii) the function the client needs to invoke once a message from the server is received and (iv) determining a stopping condition from the training service. The fusion server side functions include (i) the initialization of the routine at the fusion server and (ii) the steps to be taken on every request from a client.

These steps can be viewed as an abstract implementation of a fusion algorithm, a policy generator, or a statistics collector – depending on the phase in which the system is operating.

V. Algorithms

In this section, we explore the different algorithms that need to be implemented in each of the libraries described in Fig. 5 with more details on the policy generation library and the fusion algorithms.

A. Policy Generation

In order to deal with schema differences, the data available at each of the sites needs to be converted into a single common format over which federated learning can be performed. The exact algorithm depends on the nature of training data.

For training data belonging to category II and IV, the basic algorithm consists of finding out the ability of each of the participating training services to convert among different possible formats. In the description of the schema at each training service, the training service provides the default format of the training data, along with the alternative formats that the training service is capable of converting the data to. Thus, the fusion service gets a bipartite directed tree from each of the training services. The fusion service superimposes all those trees, and finds a common format to which all of the available training data can be converted to.

The algorithm is shown schematically for the case of three sites in Fig. 6. The first site has training data as images in .png format, and has the ability to convert them into .jpg and .bmp format. The second site has images in .bmp format and can also convert them into .png format. The third site has data in .png format, and can convert them into .png format. The algorithm draws a graph of each site and from that draws a line to the format that the site has the ability to format into. The format that has the maximum number of inbound arrows is chosen as the format of the canonical format to be used, that in this case is going to be .png format. In case there are multiple such formats, the one that requires the least number of conversions is chosen.

Another type of policy that needs to be generated is that for converting different features that may be called different names at different sites. These policies would be of the format if col. name = xyz, then rename it as abc. The mapping from the different column names of each site to a canonical format needs to be determined. In order to do so, the fusion service collects the description of the features

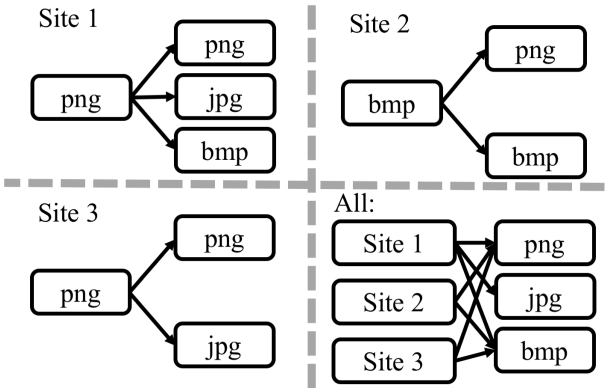

Fig. 6. Determining the common format for raw data.

from each of the training sites. It also collects a small sample of the data for each feature. This could be synthetic if the sites are under strict prohibition not to send any real data across.

The policy generation algorithm needs to be looked upon as an assisting service, as opposed to a fully automated service. In other words, the interaction flow shown in 4 needs a human check at each site once the policies are received in response to the call to `get_policy` in order to proceed. This is one of the advantages of implementing the interaction as a web-service in that the manual checks can be interspersed easily along with a automated exchanges.

The mapping of the features is done initially by comparing the feature names provided in the schema of two different sites. By default, we assume that the schema of the first site joining into a training session is considered the common schema to be used. Each subsequent site that joins the training session needs to use a set of policies to translate its local schema to that of the common schema. The initial feature name mapping is done by matching each of the names to a hashed vector from a corpus of documents, with each feature name being mapped to the name in the common schema that is the closest in the hashed vector representation. For the case where a sample is available, a document hashing approach [14] is used to calculate the closest vector.

A third type of policy that needs to be generated is the policy scheme that maps different labels in the output corpus among each other, i.e. to deal with the situation where the output labels are different at different sites. In order to deal with this approach, an AI model that is trained for classification into the labels defined for the first training site is used. Each site uses this model to classify its own feature data, and compares the label provided by the model with the original label, creating a matrix that counts the output labels of the common model against the labels of the original data. A policy that matches the label in the original data to the most frequent label output by the model is generated.

B. Statistics Aggregation and Configuration Generation

The statistics aggregation algorithms are relatively straight-forward, and consist of each site computing its statistics about the training data. This would include the

number of data points at each location, and when class labels are present, then number of data points present at each of the labeled classes. These statistics are aggregated at the Fusion Service site to get the aggregate statistics.

The statistics are used to generate the control and configuration information for the fusion of AI models. There are many different approaches for fusion of AI models, and they generally require the determination of synchronization points for the fusion of models trained at different sites. The system computes the batch size at each of the nodes so that it can be synchronized periodically in a manner that the sites can complete one pass through the entire data set at each site in approximately the same amount of time. This allows the execution of fusion algorithms to proceed in a manner that minimizes idle waiting tine.

The configuration selection process is another step where the web-service model allows for an easy interaction with a human user. The Fusion Service system can generate the configuration parameters that can be checked by a human before proceeding with the actual model fusion.

C. Model Fusion and Validation Algorithms

The exact nature of the model fusion algorithm depends on the type of AI model that is being trained. Several flavors of these algorithms are described in existing work [2], [4], [15], and these are used to perform the actual model fusion.

The model validation is a relatively straight-forward approach whereby a shared percentage of data at each site is reserved for validation. Once the model is completely trained, the system compares the performance of the fused model against each of the validation data, and combines the result to get the performance overall.

VI. Conclusions

In this paper, we have looked at a web-services based implementation of Federated Learning which has been implemented in an enterprise context. Due to the security requirements and traditional implementation of the network infrastructure of enterprise environments, along with a need to intermix human input with automated operations, a web-services based implementation of the federated learning is most suitable for use in the enterprise context. We have defined the set of services required for the federated learning implementation, and briefly described the algorithms that are used in the implementation of the system.

In our solution, we have used generative policies to deal with the challenges of varying data quality, and linguistic and lexicon mismatch; used a statistical exchange mechanism to deal with skewed classes and features, and a centralized control approach to deal with issues of size differences, access issues and trust, and engineered a library-focused approach to handle variations in different enterprise environments.

While the current set of algorithms are adequate to deal with many common encountered in the enterprise environments, the current implementation leaves many other situations unaddressed. Specifically, we need to develop algorithms to handle semantic differences in the data at different sites; algorithms for fusing AI models with different architectures; schemes to determine the trustworthiness of different sites and their models; and address issues arising from overlapping training sets between sites. Furthermore, instead of the current implementation based on a traditional web-services framework, we are exploring a micro-services oriented implementation where each of the services can be implemented in an independent instance.

Despite these limitations, the current web-services based implementation provides a basic framework that can be used to address a broad set of problems within the enterprise, and addresses the issues which prevent easy movement of data.

References

[1] G. Cirincione and D. Verma, "Federated machine learning for multi-domain operations at the tactical edge," in *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications*, vol. 11006. International Society for Optics and Photonics, 2019.

[2] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan, "When edge meets learning: Adaptive control for resource-constrained distributed machine learning," in *IEEE International Conference on Computer Communications*, 2018.

[3] H. T. Vo, M. Mohania, D. Verma, and L. Mehedy, "Blockchain-powered big data analytics platform," in *International Conference on Big Data Analytics*. Springer, 2018, pp. 15–32.

[4] D. Verma, S. Julier, and G. Cirincione, "Federated ai for building ai solutions across multiple agencies," in *AAAI FSS-18: Artificial Intelligence in Government and Public Sector, Arlington, VA, USA*, 2018.

[5] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical secure aggregation for federated learning on user-held data," *arXiv preprint arXiv:1611.04482*, 2016.

[6] H. B. McMahan, E. Moore, D. Ramage, S. Hampson *et al.*, "Communication-efficient learning of deep networks from decentralized data," *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2017.

[7] S. J. S. P. S. C. Dinesh C. Verma, Graham White and G. Cirincione, "Approaches to address the data skew problem in federated learning," in *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications*, vol. 11006. International Society for Optics and Photonics, 2019.

[8] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[9] T. Nelson, *Mergers and Acquisitions from A to Z*. Amacom, 2018.

[10] T. Pham, G. Cirincione, A. Swami, G. Pearson, and C. Williams, "Distributed analytics and information science," in *2015 18th International Conference on Information Fusion (Fusion)*. IEEE, 2015, pp. 245–252.

[11] D. Verma, S. Calo, S. Witherspoon, I. Manotas, E. Bertino, A. M. A. Jabal, G. Cirincione, A. Swami, G. Pearson, and G. de Mel, "Self generating policies for machine learning in coalition environments," in *Policies for Autonomic Data Governance at ESORICS*, 2018.

[12] S. Har-Peled, D. Roth, and D. Zimak, "Maximum margin coresets for active and noise tolerant learning." in *IJCAI*, 2007, pp. 836–841.

[13] A. Holovaty and J. Kaplan-Moss, *The definitive guide to Django: Web development done right*. Apress, 2009.

[14] S. SrirangamSridharan, M. Srivatsa, R. Ganti, and C. Simpkin, "Doc2img: A new approach to vectorization of documents," in *2018 21st International Conference on Information Fusion (FUSION)*. IEEE, 2018, pp. 2172–2178.

[15] D. Verma, S. Chakraborty, S. Calo, S. Julier, and S. Pasteris, "An algorithm for model fusion for distributed learning," in *Ground/Air Multisensor Interoperability, Integration, and Networking for Persistent ISR IX*, vol. 10635. International Society for Optics and Photonics, 2018, p. 106350O.