# Federated Learning

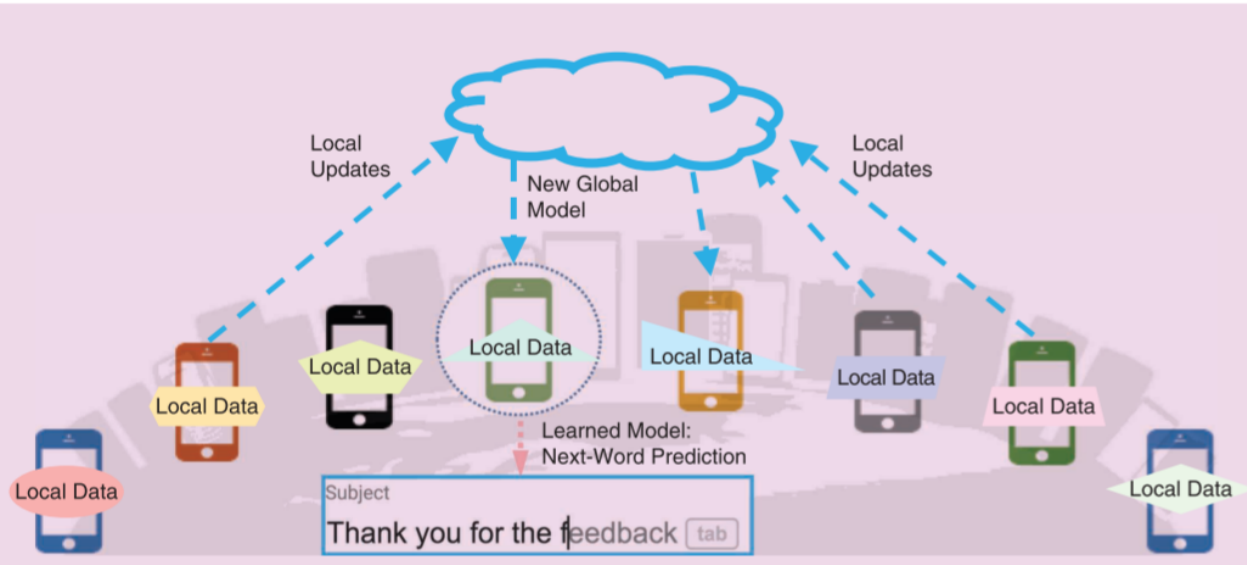
> Author @Süleyman GÖLBOL

Federated learning is used for training statistical models over remote devices or siloed data centers, such as mobile phones or hospitals, while keeping data localized.

It works by training machine learning models centrally but serving and storing them locally
(for example, this is a common approach in mobile user modeling and personalization)

Also it’s possible to leverage enhanced local resources on each device.

Training statistical models directly on remote devices. (communication network, such as nodes, clients, sensors, or organizations.) It does that without diminishing the user experience or leaking private information.




Mathematicsal formula for Problem formulation (purpose is minimize the objective function)

$$\min_{w} F(w), \text{ where } F(w) := \sum_{k=1}^{m} p_k F_k(w). \tag{1}$$

Here, $m$ is the total number of devices, $p_k \geq 0$ and $\Sigma_k p_k = 1$, and $F_k$ is the local objective function for the $k$th device. The local objective function is often defined as the empirical risk over local data, i.e., $F_k(w) = 1/n_k \Sigma_{j_k=1}^{n_k} f_{j_k}(w; x_{j_k}, y_{j_k})$, where $n_k$ is the number of samples available locally. The user-defined term $p_k$ specifies the relative impact of each device, with two natural settings being $p_k = (1/n)$ or $p_k = (n_k/n)$, where $n = \Sigma_k n_k$ is the total number of samples. We will reference (1) throughout the article but, as discussed in the next section, we note that other objectives or modeling approaches may be appropriate depending on the application of interest.

## Communication Efficiency

1) local updating methods, 2) compression schemes, and 3) decentralized training.

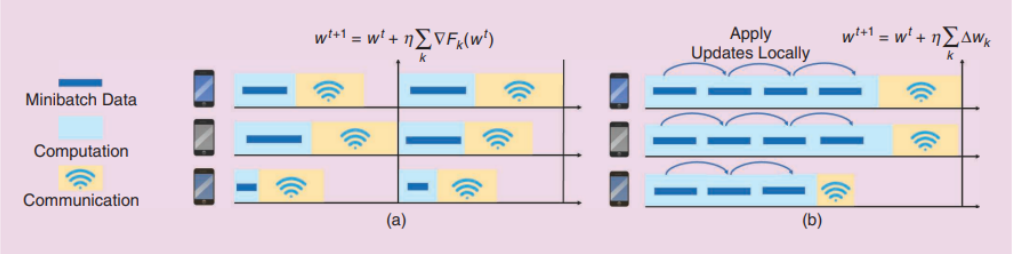



**FIGURE 2.** (a) The distributed (minibatch) SGD. Each device, $k$, locally computes gradients from a minibatch of data points to approximate $\nabla F_k(w)$, and the aggregated minibatch updates are applied on the server. (b) The local updating schemes. Each device immediately applies local updates, e.g., gradients, after they are computed, and a server performs a global aggregation after a (potentially) variable number of local updates. Local updating schemes can reduce communication by performing additional work locally.

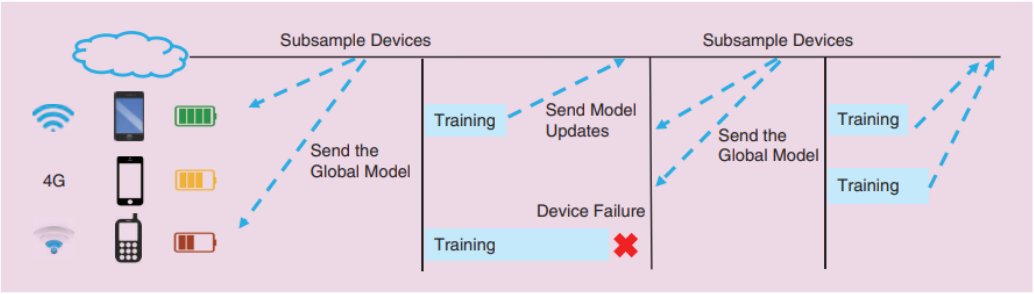



**FIGURE 4.** Systems heterogeneity in federated learning. Devices may vary in terms of network connection, power, and hardware. Moreover, some of the devices may drop at any time during training. Therefore, federated training methods must tolerate heterogeneous systems environments and low participation of devices, i.e., they must allow for only a small subset of devices to be active at each round.

If we pull a device from server, the staleness depends on the number of other devices that have updated. Even though asynchronous parameter servers have been successful in distributed data centers, classical bounded-delay assumptions can be unrealistic in federated settings, where the delay may be on the order of hours to days or completely unbounded.

Due to the decentralized approach FL has, we do not need to worry about actively training the algorithms ourselves very much. The algorithms we use for FL train themselves directly on the devices and only transfer back the relevant data that is needed.

**Work Logic**

FL is a distributed collaborative AI approach that allows for data training by coordinating multiple devices with a central server without sharing actual datasets [6]. For instance, multiple IoT devices can act as workers to communicate with an aggregator (e.g., a server) for performing neural network training in intelligent IoT networks. More specifically, the aggregator first initiates a global model with learning parameters. Each worker downloads the current model from the aggregator, computes its model update, e.g., via stochastic gradient descent (SGD), by using its local dataset, and offloads the computed local update to the aggregator. Then, the aggregator combines all local model updates and constructs a new improved global model. By using the computing power of distributed workers, the aggregator can enhance the training quality while minimizing user privacy leakage. Finally, the local workers download the global update from the aggregator, and compute their next local update until the global training is complete.

## Statistical heterogeneity

Challenges arise when training federated models from data that are highly nonidentically distributed across devices, both in terms of modeling heterogeneous data and in terms of analyzing the convergence behavior of associated training procedures.

## Modeling heterogeneous data

There exists a large body of literature in machine learning that models statistical heterogeneity via methods such as metalearning and multitask learning; these ideas have been recently extended to the federated setting. For instance, MOCHA, an optimization framework designed for the federated setting, can allow for personalization by learning separate but related models for each device, while leveraging a shared representation via multitask learning. This method has provable theoretical convergence guarantees for the considered objectives but is limited in its ability to scale to massive networks and is restricted to convex objectives. Another approach models the star topology as a Bayesian network and performs variational inference during learning. Although this method can handle nonconvex functions, it is expensive to generalize to large federated networks. Provably metalearn a within-task learning rate using multitask information (where each task corresponds to a device) and demonstrate improved empirical performance over vanilla FedAvg.

---

## Privacy in machine learning

The three main strategies in privacy-preserving machine learning, each of which are briefly reviewed, include differential privacy to communicate noisy data sketches, homomorphic encryption to operate on encrypted data, and secure function evaluation (SFE) or multiparty computation.

- Among these various privacy approaches, **differential privacy** is most widely used due to its strong information theoretic guarantees, algorithmic simplicity, and relatively small systems overhead. Simply, a randomized mechanism is differentially private if the change of one input element will not result in too much difference in the output distribution; this means that one cannot draw any conclusions about whether or not a specific sample is used in the learning process. Such sample-level privacy can be achieved in many learning tasks. For gradient-based learning methods, a popular approach is to apply differential privacy by randomly perturbing the intermediate output at each iteration. Before applying the perturbation, e.g., via binomial noise, it is common to clip the gradients to bound the influence

- Beyond differential privacy, homomorphic encryption can be used to secure the learning process by computing on encrypted data, although it has currently been applied in limited settings, e.g., training linear models. When the user-generated data are distributed across different data owners, another natural option is to perform privacy-preserving learning via SFE or SMC. The resulting protocols can enable multiple parties to collaboratively compute an agreed-upon function without leaking input information from any party except for what can be inferred from the output. Thus, although SMC cannot guarantee protection from information leakage, it can be combined with differential privacy to achieve stronger privacy guarantees. However, approaches along these lines may not be applicable to large-scale machine

Current works that aim to improve the privacy of federated learning typically build upon previous classical cryptographic protocols such as SMC

---

## Most commonly studied communication schemes in distributed optimization

These are **bulk synchronous approaches** and **asynchronous approaches** (where it is assumed that the delay is bounded). These schemes are more realistic in data center settings, where worker nodes are typically dedicated to the workload, i.e., they are ready to “pull” their next job from the central node immediately after they “push” the results of their previous job. In contrast, in federated networks, each device is often undedicated to the task at hand and most devices are not active on any given iteration. Therefore, it is worth studying the effects of this more realistic device-centric communication scheme, in which each device can decide when to “wake up,” at which point it pulls a new task from the central node and performs some local computation.

## Heterogeneity diagnostics questions

Recent works have aimed to quantify statistical heterogeneity through metrics such as local dissimilarity but these metrics cannot be easily calculated over the federated network before training occurs. So there are 3 open questions:

1) Do simple diagnostics exist to quickly determine the level of heterogeneity in federated networks a priority?
2) Can analogous diagnostics be developed to quantify the amount of systems-related

heterogeneity?
3) Can current or new definitions of heterogeneity be exploited to design new federated optimization methods with improved convergence, both empirically and theoretically?

## What FL Attempts

FL simply attempts to answer this main question if can we train the model without needing to transfer data over to a central location. Within the FL framework, the focus is geared towards collaboration, which is not always achieved through standard machine learning algorithms. In addition, FL allows the algorithms used to gain experience, which is also something that cannot always be guaranteed through traditional machine learning methods. FL has been employed in a variety of applications, ranging from medical to IoT, transportation, defense, and mobile apps.

---

**General architecture of FL figure**

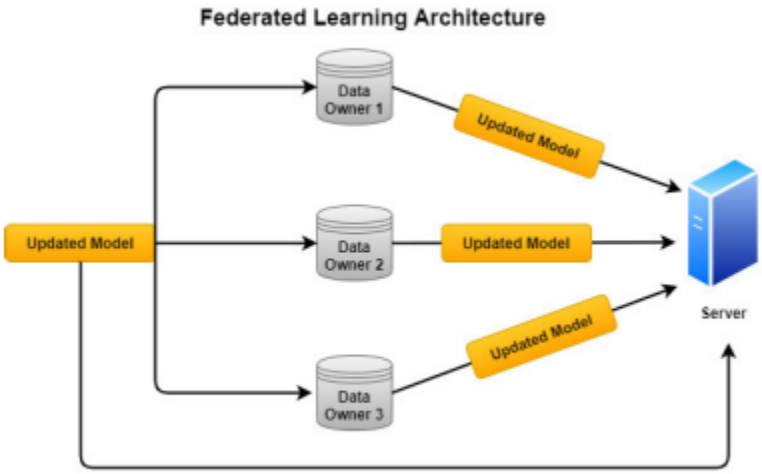



**FIGURE 1. General federated learning architecture.**

## Comparisons with other

### FL vs Traditional ML

Traditional machine learning usually has a main server that handles data storage and training models. Typically, there are two ways of using these trained models of machine learning. Either we build a pipeline for the data so it can pass through the server, or transfer the machine learning models to any device that interacts with the environment.

Unfortunately, both of these approaches are not optimal because their models are not able to rapidly adapt. With FL, the models get trained at the device level. So the models are brought over to the data sources or devices for training and prediction [16]. The models (i.e., models' updates) are sent back to the main server for aggregating. Then, one consolidated model gets transferred back to the devices using concepts from distributed computing [17]. This is so that we can track and re-distribute each of the models at various devices. FL's approach is very advantageous for utilizing low-costing machine learning models on devices such as cell phones and sensors.

While Machine Learning does allow us to pre-process data to reduce latency, it can still lead to high monetary costs. For communication, because the data can be highly independent and highly distributed, this can create some inter-operability issues with Machine Learning algorithms using the data, because the data could end up not being useable.

## FL vs DNN

DNNs are not always optimal for FL because as data-sets increase, so do the DNNs' complexity. So the DNN's complexity is proportional to the computational requirements and memory demands. Additionally, Deep Learning models applied for FL need to be able to still work without accessing raw client data, so privacy is the main focus. Comparisons of deep learning models have been made in regard to the protection offered, performances, resources, etc.

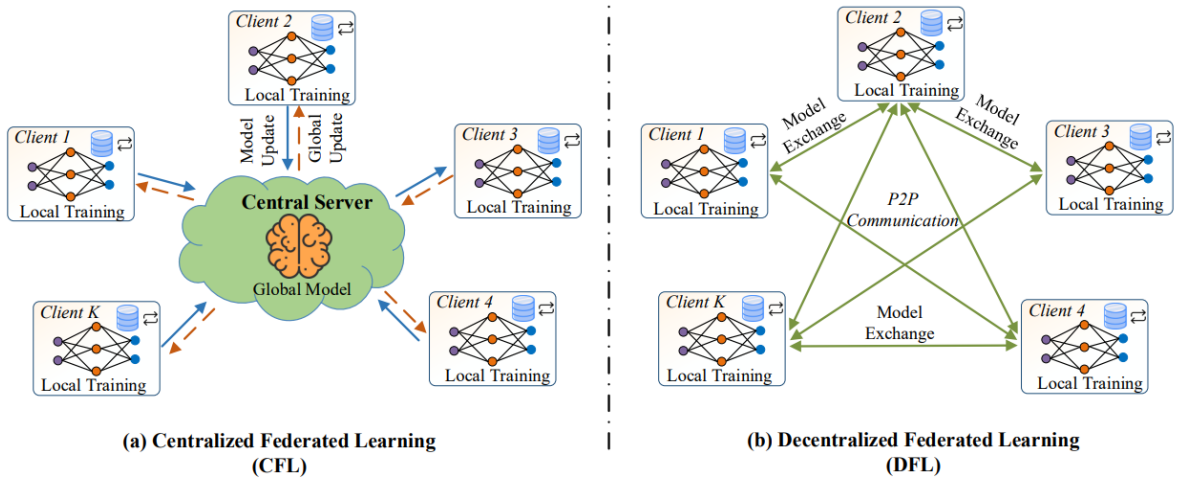



Fig. 4: Types of FL models with networking structure.

## Horizontal FL

Google proposed a Horizontal (feature based) FL approach for managing the android mobile phone updates. From a more technical perspective, Horizontal FL assumes there are honest consumers and security against a server. So just the central server can modify consumers' data. Also, Horizontal FL has been used in medical cases, such as drug detection.

In HFL systems, all learning clients cooperatively train a global FL model using their local datasets with the same feature space but different sample space. Due to the same data feature, clients can use a same AI model (e.g., linear regression, SVM) for their local training. In a HFL system, each client locally trains its AI model to compute a local update. For improved security, the computed local update can be masked by using encryption or differential privacy techniques. Then, the server aggregates all local updates from clients and computes the new global update without the need for direct access to local data. Finally, the server sends back the global update to all clients for the next round of local learning.

### Vertical FL

Vertical FL, what we are doing is collecting and grouping these various features. Then we need to calculate training loss so we can form a model that contains data from both entities collaboratively. Under Vertical FL, each entity has the same identity and status.

There are two main parts in the Vertical FL architecture: (1) Encrypted entity alignment and (2) Encrypted model training. A benefit of this architecture is that it is independent of other machine learning methods.

In VFL, an entity alignment approach is adopted to collect overlapped data samples of clients. These samples are combined to train a common AI model using encryption techniques. Example: VFL in IoT applications can be the shared learning model among entities in a smart city.

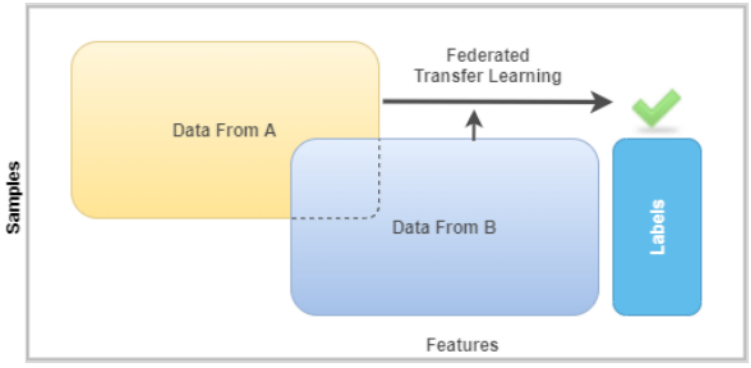
### FTL (Federated Transfer Learning)

With FTL, it is used to utilize data from a different source for training the models. With transfer learning, it involves learning a common representation between one entity's features and reducing error in predicting the labels for the targeted entity. That way, accuracy loss is minimal. FTL obtained huge attention in various industries, especially healthcare. To avoid potentially exposing client data, FTL utilizes encryption and approximation to make sure privacy is in fact, protected. So as a result, the actual raw

data and models are both kept locally [40]. There are also three components of the FTL system:

1. **Guest - Data holder:** Guests are responsible for launching task-based and multi-party model training with data-sets provided by both itself and the Host. They mainly deal with data encrypting and computation.
2. **Host:** Also a Data holder.
3. **Arbiter:** Sends public keys to both the Guest and Host. The Arbiter is primarily responsible for collecting gradients and check whether the loss is converging.

### FTL Framework




**E 5. Framework of federated transfer learning.**

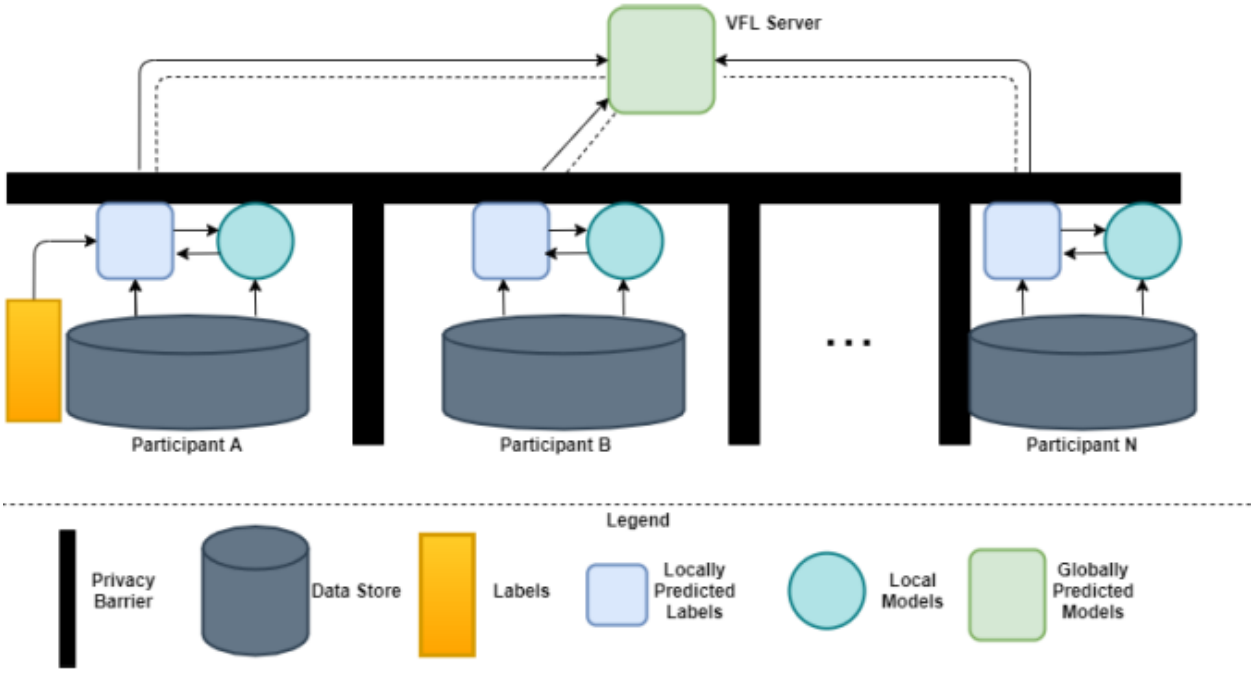
The FTL framework continues iteratively until the loss function converges.

FTL was meant to replace deep-learning approaches since deep-learning approaches are prone to accuracy loss. The authors that proposed FTL were able to conclude that FTL's accuracy is higher. The authors of the proposed framework were able to conclude that FTL is more scalable and flexible. Unfortunately, FTL does have some limitations. For instance, it needs entities to exchange encrypted results from only the common representation layers. So they do not apply to all transfer mechanisms.

### MMVFL (Multi-Participant Multi-class Vertical Federated Learning framework)

A new architecture based on the Vertical FL system. Specifically, the authors' proposed architecture is called the Multi-Participant Multi-class Vertical Federated Learning framework (MMVFL). This particular framework, shown in Figure below, is supposed to handle multiple participants. MMVFL enables label sharing from its owner to other

participants in a manner of privacy preservation. One problem with using Horizontal FL architecture is the assumption that data-sets from different entities have the same feature area yet they may not similar to the same sample ID space. That is not always the case, unfortunately, so the proposed framework is meant to alleviate that setback. With the MMVFL framework, the goal is to learn multiple frameworks to complete different various objectives. The reason for doing so is to make the learning process more personalized. To evaluate the framework's performance, the authors used two computer-vision data-sets. The authors also compare their framework against other methods. Their framework achieved better results depending on how many features used. The MMVFL framework was also noted to perform better by using a smaller amount of features as well

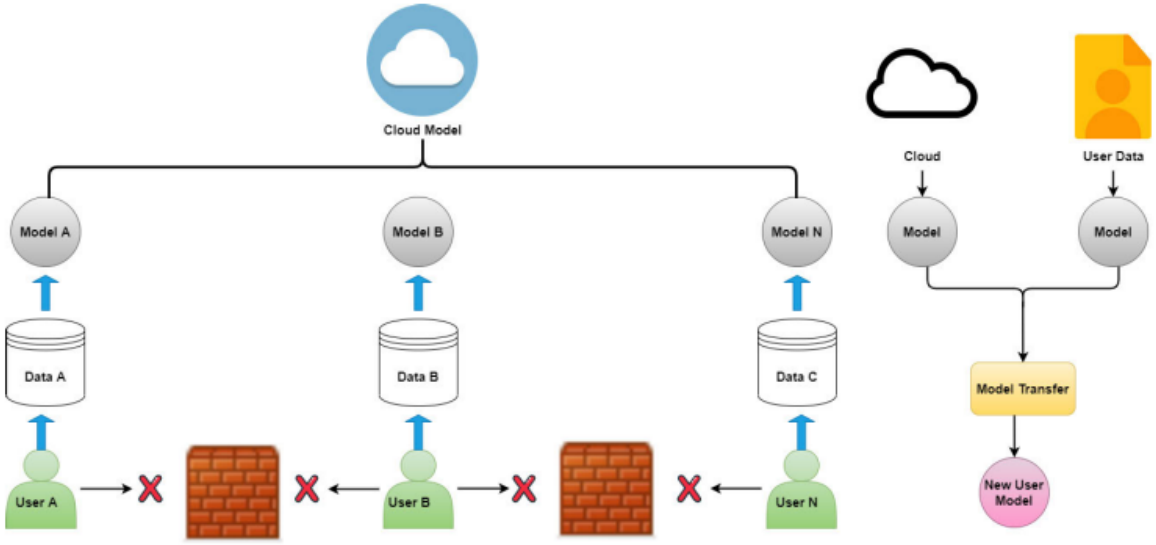



## FEDF

In terms of FL, geared toward privacy preservation and parallel training. FEDF framework was tested on different datasets, mainly the CIFAR-10 membrane data-set named MEMBRANE, and a medical image data-set called HEART-VESSEL. The evaluation metrics used were training speed, performance, and amount of data exchanged. Upon experimentation, the authors were able to conclude their proposed framework was able to improve training speed nine times faster without sacrificing accuracy.

## Smart Healthcare

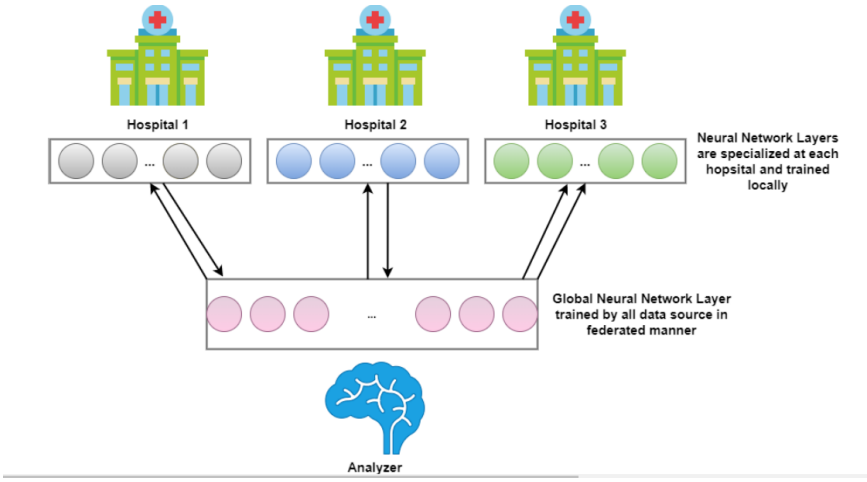
This FL architecture geared towards a specific industry comes from an article, the authors propose an FL framework centering around healthcare, specifically wearable devices, which fall into the category of Smart Healthcare. The proposed architecture is called FedHealth, shown in Fig. According to the authors, the challenges that Smart Healthcare is faced with are a lack of personalization and how to access user data without violating privacy.




## FADL (Federated-Autonomous Deep Learning)

Another architecture geared towards the medical field and this architecture is called Federated-Autonomous Deep Learning (FADL), shown in Figure. This architecture dealt with Electronic Health Records (EHR). EHR data is usually collected via individual institutions and stored across locations.  can be difficult and slow thanks to security, privacy, regulatory, and operational setbacks. According to the authors, the FADL framework trains some parts of a model using all data sources plus additional parts using data via certain data resources. To test their framework, the authors used ICU hospital data. The authors also used data from fifty-eight hospitals. Regarding FADL's structure, the structure consists of an artificial neural network (ANN) with three layers. Upon testing, the authors were able to conclude that FADL managed to out-perform traditional FL.

**Framework of FADL**




## Libraries for FL

- PySyft
- LEAF
- CrypTen
- Functional Federated Learning in Erlang (FFL-ERL)
- Tensor/IO
- FATE (Federated AI Technology Enabler)
- Tensor Flow Federated (TFF)

## Optimization techniques for FL Models

**TABLE 4. Optimization techniques to federated learning models.**

| Technique | Advantage(s) |
|---|---|
| Mini-batch Gradient Descent | Efficiency<br>Frequent Model Updates |
| Ada-grad | Faster<br>More Reliable |
| RMSProp | Low Memory Requirements |
| Fed Adagrad | Adaptability |
| FedYogi | Adaptability |
| FedAdam | Higher Accuracy |

Optimization methods FedAdaGrad, FedYogi, and FedAdam was done in Tensor Flow Federated (TFF). The three optimization methods were compared against the standard FedAvg algorithm. The algorithms managed to have higher accuracy than the standard FedAvg algorithm.

**TABLE 5. Optimization techniques to Federated Learning models.**

| Network Protocol | Focus | Benefit(s) |
|---|---|---|
| Hybrid FL | Communication<br>Resource Scheduling<br>Accuracy | Accuracy |
| FedCS | Clients' training process | Higher accuracy<br>Robust models |
| PrivFL | Mobile Networks<br>Data Privacy | Accounts for threats |
| VerifyNet | User privacy<br>Integrity | Receptiveness<br>High security |
| FedGRU/JAP | Traffic flow prediction | Reduced overhead |

## Various challenges of FL

- Missing Features (Entity has training data contains features don't exist in other entity's  data)
- Data Imbalance
- Missing Classes
- Missing Values
- Security and Privacy Issues in F
- Communication and Learning Convergence Issues of FL-IoT
- Resource Management in FL-IoT
- Feasibility of Deploying AI Learning Functions on IoT Sensor
- Standard Specifications Support of computing services / Protocols

## Attack Disadvantage

FL is prone to a variety of attacks that can compromise the model performance and accuracy. One attack is called the Data-Poisoning attack, where a person can tamper with the model by creating poor-quality data for training that model. The goal here is to generate false parameters. These types of attacks can achieve high misclassification, up to ninety percent. There can be different modifications to the data-poisoning attack. Another attack is Free-Riding Attack, where adversary wants to leech benefits from the model without being part of the learning process. This results in legitimate entities contributing more resources to the training process.

---

## Benefits and Costs

### Benefits of FL

- Data Security and Privacy
- Real Time Prediction
- Offline Prediction
- Minimal Infrastructure

### Costs of FL

• Drop out device middle of process
• Non-independent identically distributed data (NonIID)
• Unbalanced data-sets
• Large-scale distribution

## FL Applications

▼ GOOGLE KEYBOARD QUERY SUGGESTIONS

The authors first collected training by observing how consumers interact with Gboard. The authors had to ensure the consumers data usage and user experience were not negatively impacted. So they used Android's Job Scheduler for managing background operations when the devices are idle or charging. The authors also managed to build a client-server architecture. And they were able to fully build and train their model successfully for improving keyboard search suggestions.

▼ MOBILE KEYBOARD PREDICTION / KEYWORD SPOTTING

Authors trained a Recurrent Neural Network (RNN). Their RNN was deployed on a framework FL. Consumers usually expect a visible keyboard response within about 20 milliseconds of an input event. Due to how frequently mobile keyboard apps are used, client device batteries could be rapidly depleted. The authors managed to use what is called the Coupled Input and Forget Gate (CIFG) network, which is a type from the notable LSTM (Long Short Term Memory) RNN. CIFG uses a single gate for controlling both the input and recurrent cell self-connections, thus shrinking the number of parameters by twenty-five percent. By using CIFG, the authors note that CIFG is beneficial for mobile devices due to the reduced quantity of computations and parameters. The authors used TensorFlow for training the model. In addition, FL was used for training the server-side of CIFG. This particular training relied on data from Gboard users who have opted to share bits of text while typing in Google apps. The text is modified to contain short phrases of a few words. For this application, 7.5 billion sentences are used for training, while the test and evaluation sets each have 25,000 sentences. The average sentence length in the dataset is about four words. For the FL training, the authors stored data in local caches on devices. However, in order for the clients to partake in FL, they needed to meet a certain amount of requirements. The devices need to have at least 2GB (gigabytes) of memory available. Also, the clients are only allowed to participate if they are charging, connected to a network, and idle. Upon testing, the authors were able to have their CIFG network trained on FL achieve optimal performance.

▼ RANKING BROWSER HISTORY SUGGESTIONS

FL was applied towards ranking Firefox browser history suggestions. Authors used FL to train a model based on user-interactions while focusing on privacy. For robustness andoptimization, they used a variant of the RMSProp optimization technique. FL can be difficult to deploy with real users because consumes more

time and can lead to negative user experience. When applying FL for ranking browser history suggestions, the authors deployed their system to a large number of Firefox users. In less than a week, those users were able to help train and evaluate the new model for Firefox's improved URL bar. The new model leads to users typing over half a characterless. Authors managed to prototype their work by using simulations. (also simulating FL optimization process). The authors used mock dataset and client-server components for their system. Overall, the authors' application of using FL to improve ranking browser history suggestions was optimal and managed to preserve the consumers' privacy.

▼ VISUAL OBJECT DETECTION

Authors designed a platform called FedVision in order to support FL-powered computer vision applications. Tobject detection, the users use the data obtained from cameras and uploads to a center for training. Once a model has been trained, it can be used for various tasks. However, the disadvantage here is that the users do not have control over how the data would be used. In regards to FedVision's framework, FedVision uses a visual object detection framework based on YOLOv3. FedVision also allows for the training of object detection models with data-sets stored locally via multiple clients. The user-interaction is designed so that there is no requirement for users to be familiar with FL FedVision has also managed to help consumers improve their operational efficiency, achieve data privacy protection, and reduced cost. Such an application of FL is a notable example of how FL can be fully applicable and garner a positive impact. FedVision has also managed to inspire other applications of FL in similar situations, where the authors attempt to use FL for Human Activity Recognition.

▼ PATIENT CLUSTERING TO PREDICT MORTALITY ANDHOSPITAL STAY TIME

EMRs can't easily stored and shared in centralized locations because EMRs are generated by patients in various healthcare facilities and clinics. They are sensitive, so traditional machine learning mechanisms are not applicable. There are concerns regarding the storage of EMRs, security, privacy, cost, and availability of sharing medical data. While FL can greatly tackle these issues, FL may not perform optimally depending on how the data is set up. So authors proposed an algorithm called CBFL (Community Based Federated Learning) to remedy this issue. The authors used the CBFL algorithm to predict patient mortality and stay time at the healthcare facility. CBFL was developed based on the eICU collaborative research

database, which has data of a little over two-hundred thousand patients admitted to two-hundred hospitals. There are three procedures in the CBFL algorithm, which are: encoder training, K-Means Clustering, and Community-Based Learning. Results indicated that CBFL achieved high accuracy and out-performed the standard FL model.

▼ DRUG DISCOVERY

FL has been used for drug discovery. But works tackle specific problems related to drug discovery and apply FL to solve them.

- One work focus on incorporating FL for drug discovery deals with data-sets with high biases. Biased data can be a huge problem for model training because the data itself can be skewed due to these biases. So this results in a model that may be inaccurate. The authors managed to form a general FL framework for drug discovery. The general FL framework consists of server-coordinator and collaborators for the FL client. During each round of training, the framework's server transmits the newest model to each of the clients. They handle model updates by executing training. The clients also encrypt and transfer the model updates via a protocol. Then the coordinator-server gathers model changes, then uses them to update the model. The authors managed to evaluate their proposed framework by comparing it against centralized learning and used seven drug-related data-sets. Their framework managed to outperform centralized learning.
- In other work, the focus is on using what is called the Quantitative structure-activity relationship (QSAR) and applying FL to it for drug discovery. QSAR analysis is typically used in, and having various facilities collaborate usually leads to better results. QSAR is typically used to investigate and predict various properties of compounds, which is crucial for drug discovery. Because of hindrance of intellectual property, which can hinder collaboration for drug discovery via QSAR. a new platform for FL-based drug discovery called FL-QSAR. FL-QSAR uses horizontal FL architecture. For testing, Shaoqi Chen and his team used fifteen datasets. They also made comparisons of results from incorporating horizontal FL versus not incorporating horizontal FL, collaborations via horizontal FL and single clients, and a classic privacy-preservation framework. Results indicated that FL-QSAR outperforms single

clients, and by using the horizontal FL architecture for FL-QSAR, FL-QSAR is efficient for collaboration between pharmaceutical institutions.

▼ fMRI ANALYSIS

In healthcare settings, data suffers from lack of accuracy and generalizability. High-quality data is not often available in healthcare settings due to concerns of reproducibility, models also suffer in performance. Patients are also concerned about their medical data being used for future health insurance decisions and shared with their employers. Health providers also worry that if their health statistics are made publicly available, they will lose patients or suffer huge consequences if they cannot assess their performance. Authors explore this dilemma by applying FL for functional MRI (fMRI) data. the fMRI data is related to different kinds of neurological diseases or disorders. Authors managed to form their proposed framework without data-sharing. The proposed framework consists of two main elements: local updates, and communication to a global server. For testing, the authors used resting-state fMRIs from the ABIDE dataset (Autism Brain Imaging Data Exchange). Here, they used it to identify autism spectrum disorders (ASDs) and a healthy control group. The proposed framework demonstrated the advantages of using FL and has possible use-cases for identifying rarer diseases with fewer patients.

▼ BRAIN TUMOR SEGMENTATION

This also covers medical imaging. While Deep Neural Networks have illustrated notable findings, they are very reliant on quantity and diversity of the training data and it’s problem because the needed training data may not be available due to having low incident rates of few diseases/disorders and a low number of people. So authors used the BraTS 2018 dataset, which contained MRI scans of almost 300 people with brain tumors. The authors compared their methods against data-centralized training. The results indicated the optimal performance of the authors’ proposed method.

▼ DISTRIBUTED MEDICAL DATABASES: META-ANALYSIS OF SUBCORTICAL BRAIN DATA

Purpose is accessing and analyzing biomedical data without sharing information. Focus is on analyzing brain structure relationships across various diseases. There is a lot of data containing brain images, so there is a lot of opportunities to fully

comprehend the genetic mechanisms for various brain-related diseases. Unfortunately, datasets, which are stored in unique places, cannot always be shared because of privacy and legality concerns, so we are limited in fully exploiting data for studying brain disorders. The authors' proposed FL framework was first evaluated using artificial data, then applied at multiple databases. The authors were able to validate their proposed framework was sufficient for the intended purpose.

▼ FedNER

FL for Medical Named Entity Recognition (NER). NER has many applications in a healthcare setting, but sufficient data that is labeled is crucial for training and obtaining an accurate NER model. Unfortunately, labeled data in the medical community is limited due to sensitivity. NER is focused on identifying various medical entities such as drug names, reactions, and symptoms from unstructured medical texts and classify them into different categories. The authors in this application propose an FL framework called FedNER to better make use of the medical data and obtain an accurate NER model without exchanging sensitive medical data. In the FedNER framework, the server communicates with multiple clients model updating and model sharing. Authors tested FedNER by using three medical NER data-sets, and used an 80% Training, 20% Testing. FedNER was compared against a few baseline NER methods. The authors found that their FedNER model managed to outperform other NER methods.

## FL Use-Cases

▼ MEDICAL IMAGING

Authors attempt to develop applications on a platform known as ATMOSPHERE which is a platform where users are able to develop applications. They tried to do with medical imaging. This use case also deals with federated cloud infrastructures. These are usually used for handling requests to balance workloads. As such, this allows for multiple users to handle resource demand peaks that cannot otherwise be handled locally. The use cases the authors created was for early diagnosis of Rheumatic Heart Disease (RHD) which is a cause of heart halve disease. Regarding the authors' model(figure attached) their application is an image-analysis method that handles feature extraction, which is fed into a classifier that uses deep learning mechanisms. The model is meant to differentiate between someone definitely having RHD and someone being borderline for RHD. For training, the authors used a large database that includes over four-thousand studies. These

studies contain echo-cardio videos and demographic data, and three categories are included: Definite RHD, Borderline, and Normal. For the ATMOSPHERE platform, an application contains three main components

1) **A Virtual Infrastructure Blueprint** - Here, this involves any third-party components for handling resources and back-end execution.
2) **Application Dependencies** - These are handled via docker files. The system uses these for building images.
3) **Final Application** - This is made up of the interface layer and the processing backend.

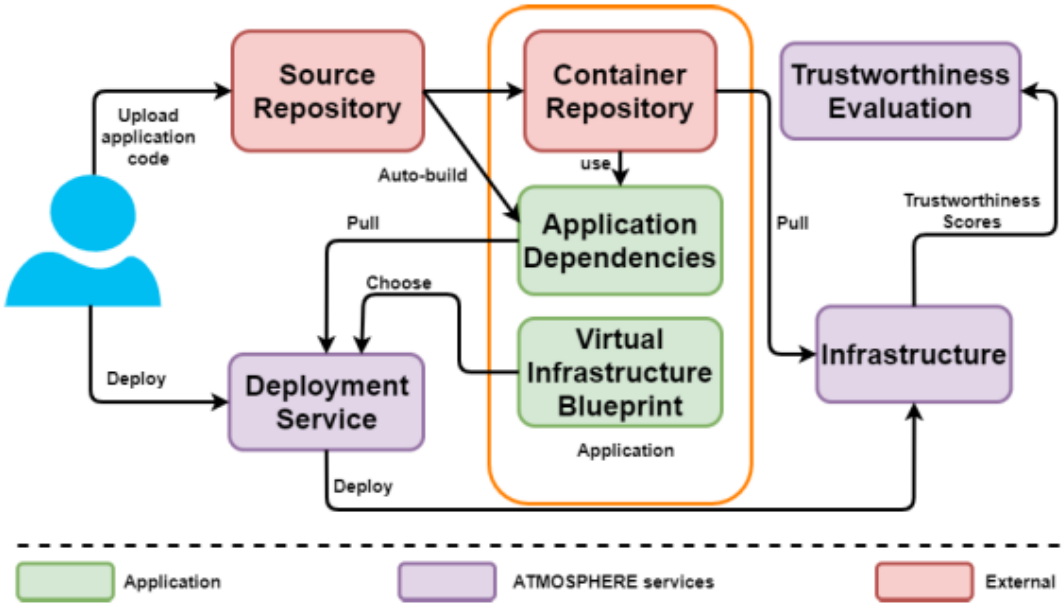



**FIGURE 18. Complete framework of ATMOSPHERE.**

▼ ANOMALY DETECTION

Authors successfully attempted to build an anomaly detection system(called DIoT) using FL to detect various IoT devices. Their system is autonomous, and can effectively operate without human intervention or labeled data. Regarding the architecture of DIoT, it consists of two components: a security gateway, and an IoT security service. The security gateway acts as a local access gateway to the

Internet. It is responsible for identifying the device's type when that device gets added into the network. The security gateway component is also responsible for performing anomaly detection for identifying any compromised devices in the network. Meanwhile, the IoT security service supports the security gateway. It maintains a repository of specific anomaly detection models depending on device type. The assumptions made are that there is no malicious manufacturers, the security gateway is not compromised, and the automated identification of IoT devices. Most notably, their system was able to achieve a 95% detection rate with no false positives. Additionally, their system is able to cope with both emerging new and unknown attacks

▼ AUGMENTED REALITY

Authors discussed high data and latency issues for handling the enormous data for augmented reality. To solve issues they proposed a framework using FL. Their framework uses a combination of FL and mobile edge computing (MEC) in order to account for the huge amounts of data and latency issues. The authors' framework treats devices as a group, mainly in charge of generating data and performing local model updates. The cloud aspect of the framework facilitates the computation of the global model depending on local models that were uploaded. The authors tested their framework's performance with the CIFAR-10 dataset, which consists of ten thousand labeled images. Their framework resulted in requiring less training, which is impressive since training models can often take a lot of time.

# FL FOR IOT SERVICES

▼ A. FL Serving as an Alternative to IoT Data Sharing

Instead of sharing the raw IoT data, FL offers an alternative of sharing learning results to enable intelligent IoT networks with low latency and privacy preservation. Due to the resource constraints of IoT users, an FL scheme is designed that enables the estimation the types of data sharing requests with queries submitted by a requester to return the correctly computed results towards these queries for sharing.

▼ FL for the Optimization of IoT Data Offloading and Caching

**Minimize Energy Cost with Video Offloading**

FL is combined with SVM to predict the user association for minimizing energy costs spent on task computation and transmission history. This can be done by using a learning federation approach with FL which allows each mobile user to learn an SVM model using its own dataset and then exchange the local updates for computing a global model based on the constraints of user association and task data size. As a result, the cooperation of users helps the BS to determine the best user association for offloading, aiming to reduce the energy consumption with high learning accuracy.

**FL-based Optimization for IoT Data Caching**:

IoT data offloaded from mobile devices can be cached by edge servers [65] where FL can play an important role in establishing intelligent caching policies, in order to cope with the explosive growth of mobile data in modern IoT networks. FL is very useful to build proactive data caching schemes in edge computing without the need for direct access to user data to predict the most popular files for caching. By using this concept, mobile users can download the NN model from a cache entity, such as an edge server, for local training before sending back the model to the server for aggregation in an iterative manner. User equipments (UEs) do not need to offload their data to the MEC servers; instead, they train the DRL model at local devices and only submit the model parameters to the server. This learning solution aims for privacy protection and spectrum resource saving.

▼ C. FL for IoT Attack Detection

Enabled by the privacy enhancement feature of FL, a federated attack detection and defense solution is built in a fashion that each industrial IoT device joins to run a DNN model locally, in order to retrain the threat model to fight against adversaries. Each IoT device first produces adversarial samples to create a retraining set; then, the local trained updates are offloaded to a cloud server for synchronization. Finally, the cloud server computes a global model and then sends back to the local devices for the next round of learning. The cooperation of multiple devices accelerates the learning process and improves the detection of adversaries while mitigating the risks of attack on the model learning

A stochastic dynamical system is designed to evaluate the aggregated parameters in each FL round by using derive explicit conditions that prevent attacks from interfering the FL process at the aggregator.

▼ D. FL for IoT Localization

For example, FL is used in to build a privacy preserving indoor localization model in residential building settings. Mobile users can build their AI models locally by using received signal strength measurements from beacons with labelled locations, while the central server builds a global multilayer perceptron model from local updates for accurate localization estimation. By using available RSS data from database, a simulation is performed, showing a significant improvement in the localization accuracy while privacy is preserved in different indoor settings. This service is also verified to solve the issues of repetitive task learning and privacy leakage risks due to the reliance of a central AI server.

▼ E. FL for IoT Mobile Crowdsensing

Traditional ML models leaves a leakage risk about privacy but FL is accelerate the learning and training for crowdsensing models. But distributed FL framework is better. Distributed FL framework in a sensing platform for distributed decision making and learning among IoT devices. To mitigate latency incurred by the central server communication, an infusing redundancy solution is considered to speed up distributed stochastic gradient descent calculation based on a distributed of datasets. In this way, each IoT node only needs to compute the model based on its mini-batch instead of the whole datasets, which will reduce computation latency accordingly.

▼ F. FL-based Techniques for Privacy and Security in IoT Services

Devices not only collect personal information but can also monitor user activities. Many AI/ML algorithms have been adopted to address security and privacy issues, by their ability to classify and detect threats and privacy bottlenecks in IoT networks. However, these traditional solutions also have several limitations, including the need for centralized IoT data collection and user privacy exposure due to public data sharing. FL appears as an attractive approach for enabling intelligent privacy and security services in IoT networks.

***For example:***, FL has been used for data privacy preservation in vehicular IoT networks where two-phase mitigating scheme is proposed for intelligent data transformation and collaborative data leakage detection. Vehicular FL solution allows participants (vehicles) to train models locally with their own data without a centralized curator, which contributes significantly to protecting their data privacy.

Moreover, a joint mapping approach is performed over multiple vehicles to ensure the utility of data after the transformation, allowing for mapping raw data of multiple parties into learned data models. The learned model contains valid information to be further used in tasks such as resource allocation without revealing their raw data, which further enhances privacy protection.

▼ G. FL-based Techniques for Privacy and Security in IoT Services

An NN-based FL scheme for network anomaly detection such that the participants do not share their training data to a third party, which can prevent the training data from being exploited by attackers. A multi-task learning method is proposed by using a distributed DNN which can perform the network anomaly detection task, traffic recognition task, and traffic classification task simultaneously.

# FL FOR IOT APPLICATIONS

Every one of these focuses on privacy.

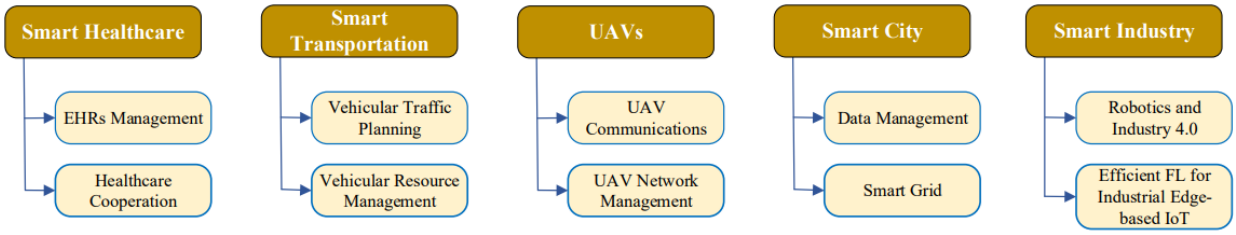



Fig. 9: FL-IoT application domains.

Click on ►button to see the context !

▼ A. FL for Smart Healthcare

FL can provide a number of efficient solutions for smart healthcare and potentially reshapes the current intelligent healthcare systems by proving AI functions for supporting healthcare services while improving user privacy and reducing low latency with the cooperation of multiple entities such as health users and healthcare providers across medical institutions. It can be learned that FL can enable flexible and privacy-preserving EHRs management in healthcare operations, by building

intelligent EHRs systems with the cooperation of multiple medical institutions and a powerful server such as the cloud.

▼ B. FL for Smart Transportation

FL can be used to replace traditional centralized ML approaches in traffic prediction tasks by running ML models directly at the edge devices, e.g., vehicles, based on their datasets such as road geometry, traffic flow and weather. The use of massive data from multiple vehicles and huge computation capability of all participant helps provide better traffic prediction outcomes, which cannot be met by using centralized ML techniques with less dataset and limited computation. Moreover, by combining with blockchain, FL is useful to build decentralized traffic planning solutions for vehicular systems

▼ C. FL for Unmanned Aerial Vehicles (UAVs)

FL can provide better learning solutions for intelligent UAVs networks by using a cooperation of multiple UAVs without transferring raw data to BSs and compromising data privacy. The use of FL would mitigate the data volume transferred over the aerial environment and thus reduce communication delays and privacy concerns, while increasing training speed of the global model due to the employment of computing resources of all UAVs. Moreover, FL enables UAVs to collaboratively train AI models only by using their partial illumination data.

▼ D. FL for Smart City

Most proposed AI-based smart city solutions rely on a centralized learning architecture on a data center, such as a cloud server, which is obviously not scalable to the rapid expansion of smart devices in smart cities. FL offers more attractive features for enabling decentralized smart city applications with high privacy levels and low communication delays. FL is also important for structuring data streams from ubiquitous IoT devices that act as FL clients for performing local learning without sharing their data with external third-parties.

▼ E. FL for Smart Industry

Coordinate the data learning among robotics for industrial tasks, e.g., traffic routing, without unpredictable network transmission delays. Edge computing is also integrated with FL to realize a collaborative learning among robotic arm devices. Due to the complexity and dynamics, each device runs a separate ML model, e.g, reinforcement learning, for determining its own control policy, and then shares

mature policy model parameters to a cloud server for aggregation. Network resource management is another critical issue in industrial FL-IoT system design that has been solved via resource allocation and stochastic training optimization.