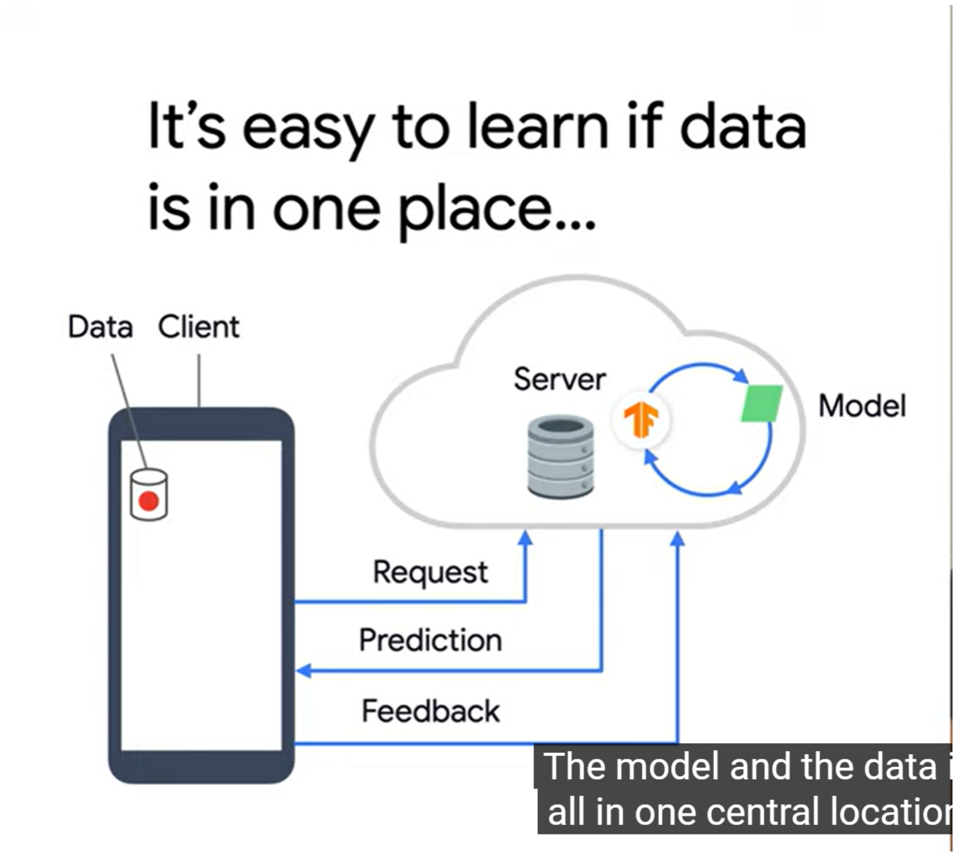
ML VS FL
FL WORKING MODEL

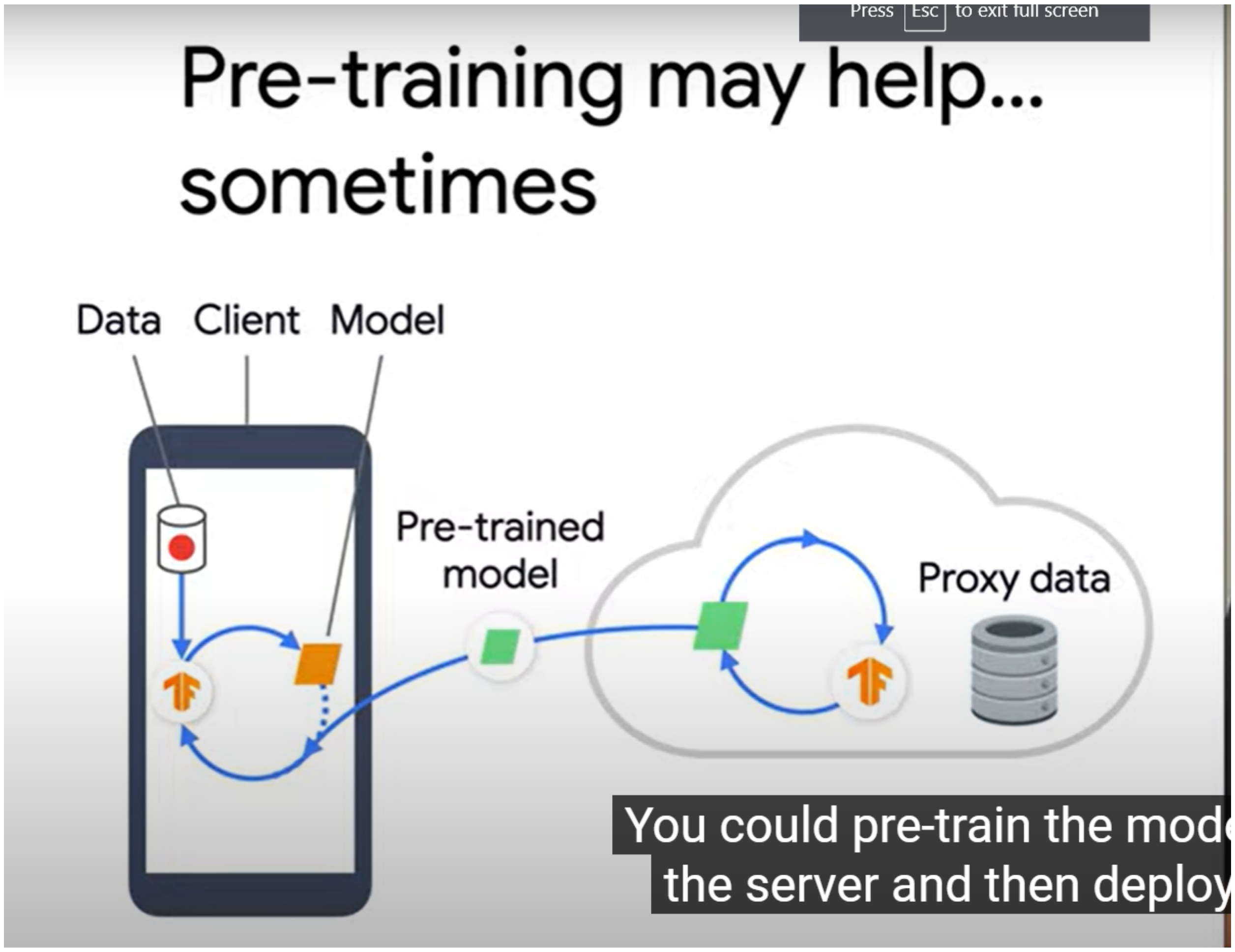

It's easy to learn if data is in one place...
Data
Client
Server
Model
Request
Prediction
Feedback
The model and the data i
all in one central locatio

ML MODEL WHICH IS USUALLY USED

The data and the model both are at the central location.
In usual ml.

- Easy usage,
- Quite an easy one, not a complex process,
- High training data,
- Centralization,
- No security,
- Privacy concerns,
- Data loss,
- Network latency,
- Connectivity,

- Battery lives,
- Power management,
- hacking.



Pre-training may help... sometimes

You could pre-train the mod
the server and then deploy

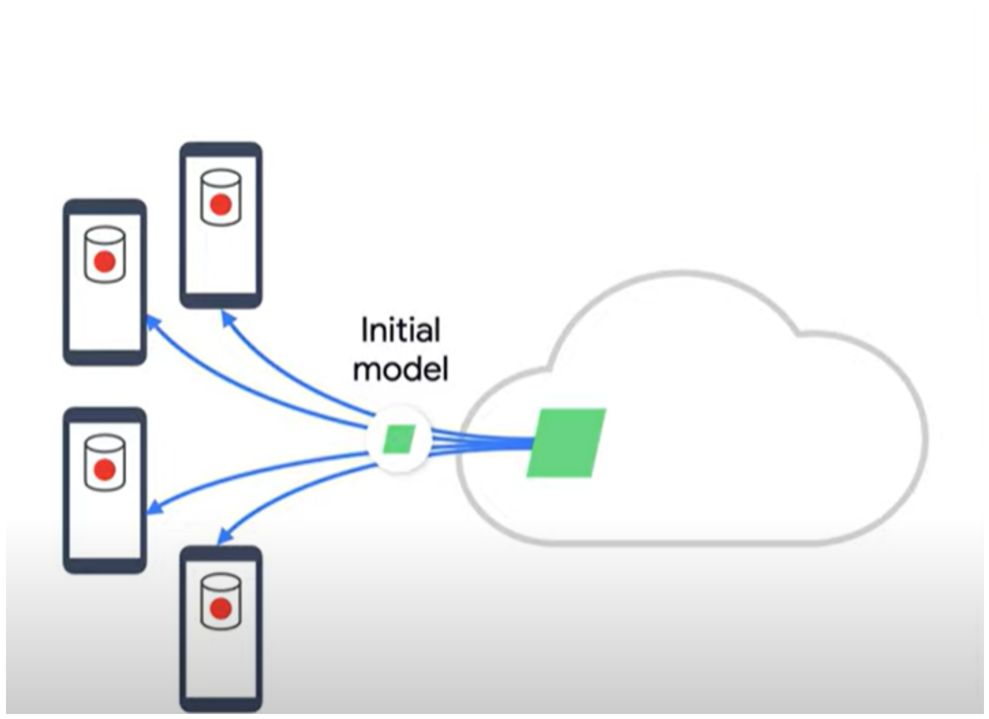
Training on the device and own data and own model, No communication necessary, We can use pretraining for this and after pretraining the model in the device we can deploy it on server or central network but this has some disadvantages like:

- Low training data,
- The model will not be perfectly trained,
- Cannot predict the behaviour with testing data,
- Overfitting with its own data and underfitting with the other data.
- If a new trendy concept comes today and the model is trained with data from yesterday it won't work well.

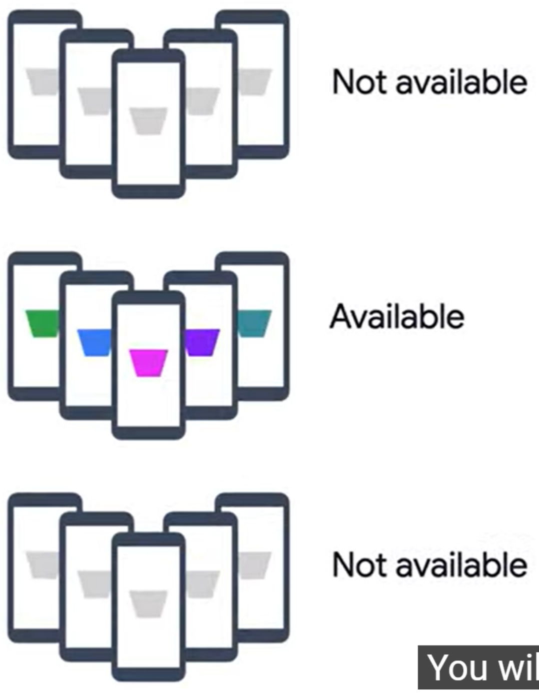
To use the goodness of decentralizing data and maintaining users' privacy, we use federated learning:
The core idea is decentralized learning(distributive learning)

User data is never sent to the central server

How it works?

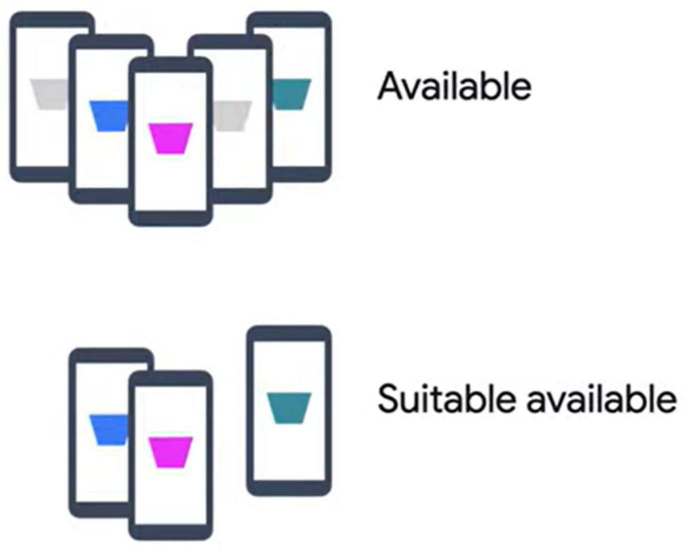


The initial model from the server will be distributed to the clients,
But this will help more effectively if we use the concept of availability and suitability,
Availability:

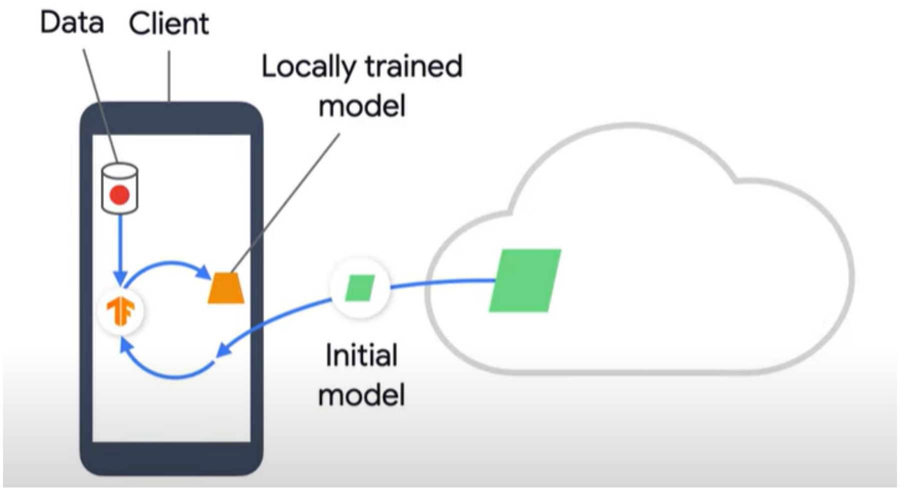


Which are plugged in are available and which are not in use are not available:
If we deploy the model to all clients it may affect the user experience.

Also, suitability, means whether the client has enough data or not, checking this would help:

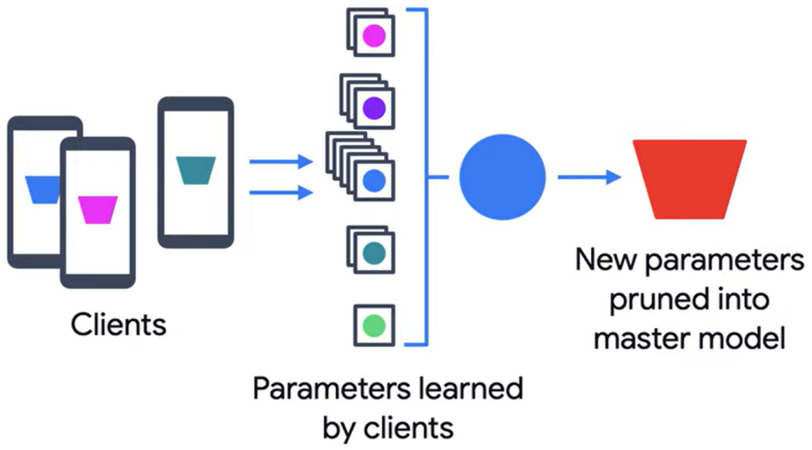


After finding suitable and available devices we deploy the model to those clients,

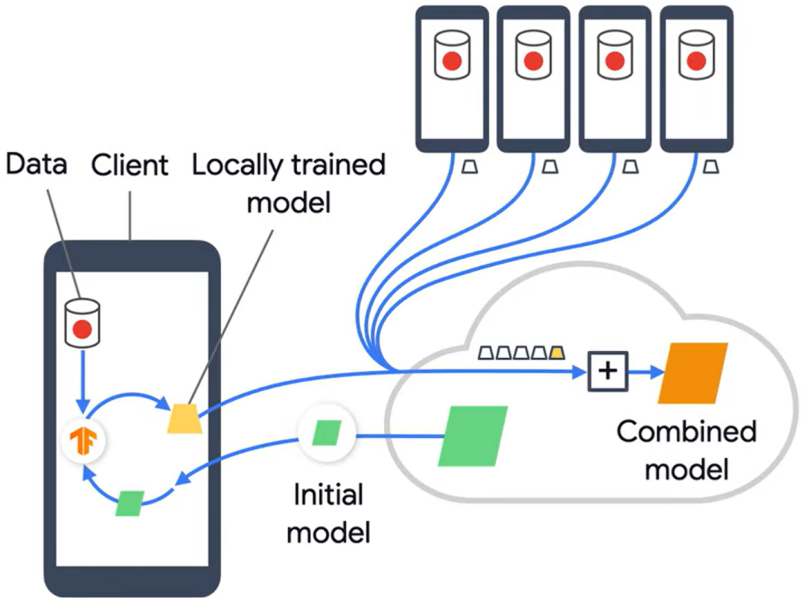


Each client trains the model locally using its own local data and produces a new model,
Which is then sent to the server,

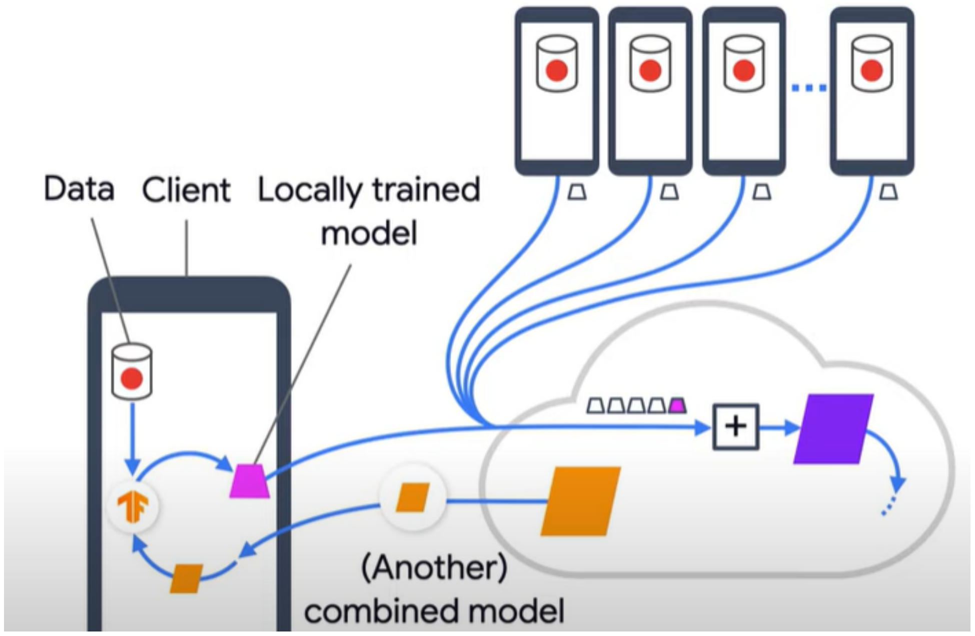
The data used to train the individual model never leaves the device,
Only the weights, bias and other parameters will leave the device,


Clients
Parameters learned by clients
New parameters pruned into master model

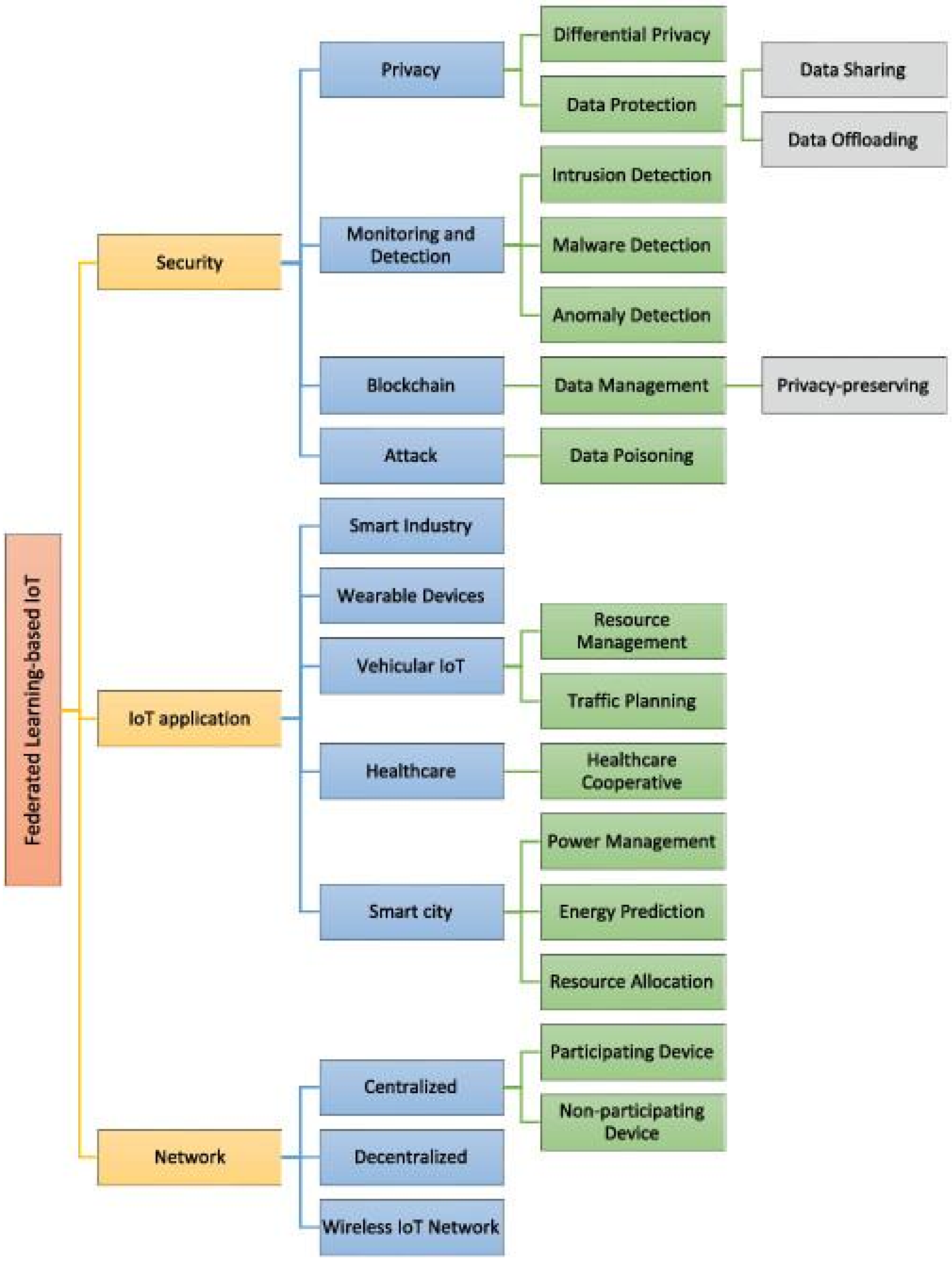
The server now gets all the locally trained models and averages them out, creating an effective master model,



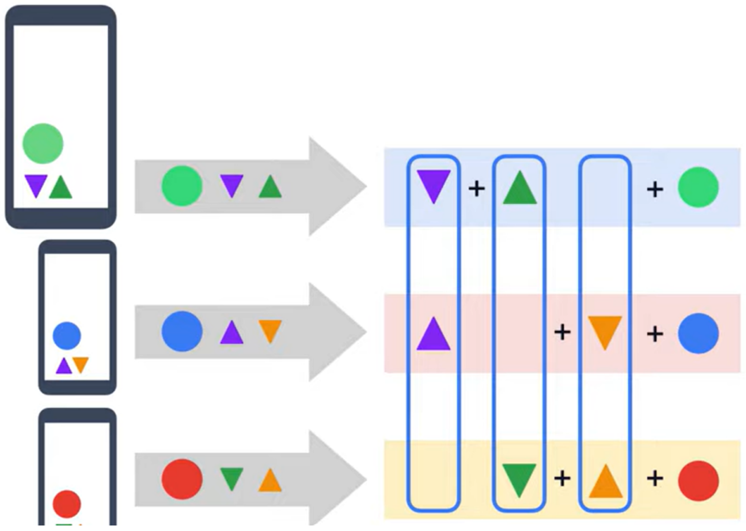
But this done once can't assure a good model, thus we need to do it again and again so that the combined model becomes the initial model for the next round, and it gets better round by round.



For additional privacy in FL we use security aggregation or encryption, even cryptography techniques can be used for the same.

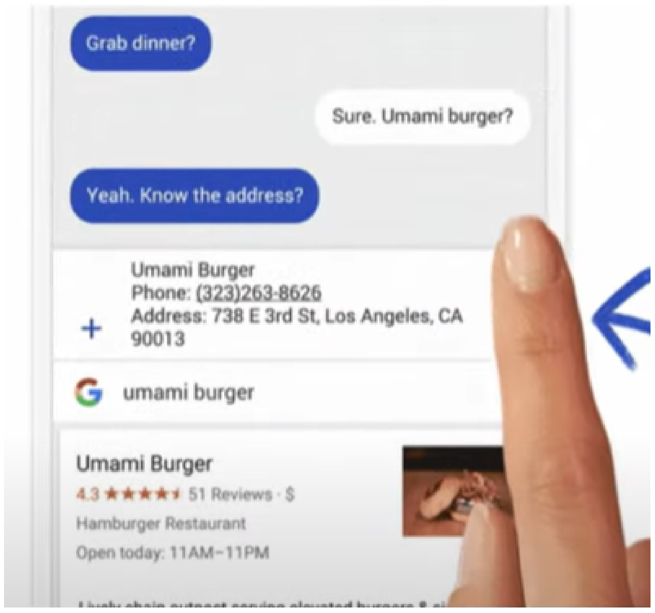


security aggregation: pairing up with buddy systems and later on it will be decrypted and only payload will be considered,



The server after getting these random values added data removes all these values as it already contains the information and reduces it to the original data.

Example of FL:



Grab dinner?

Sure. Umami burger?

Yeah. Know the address?

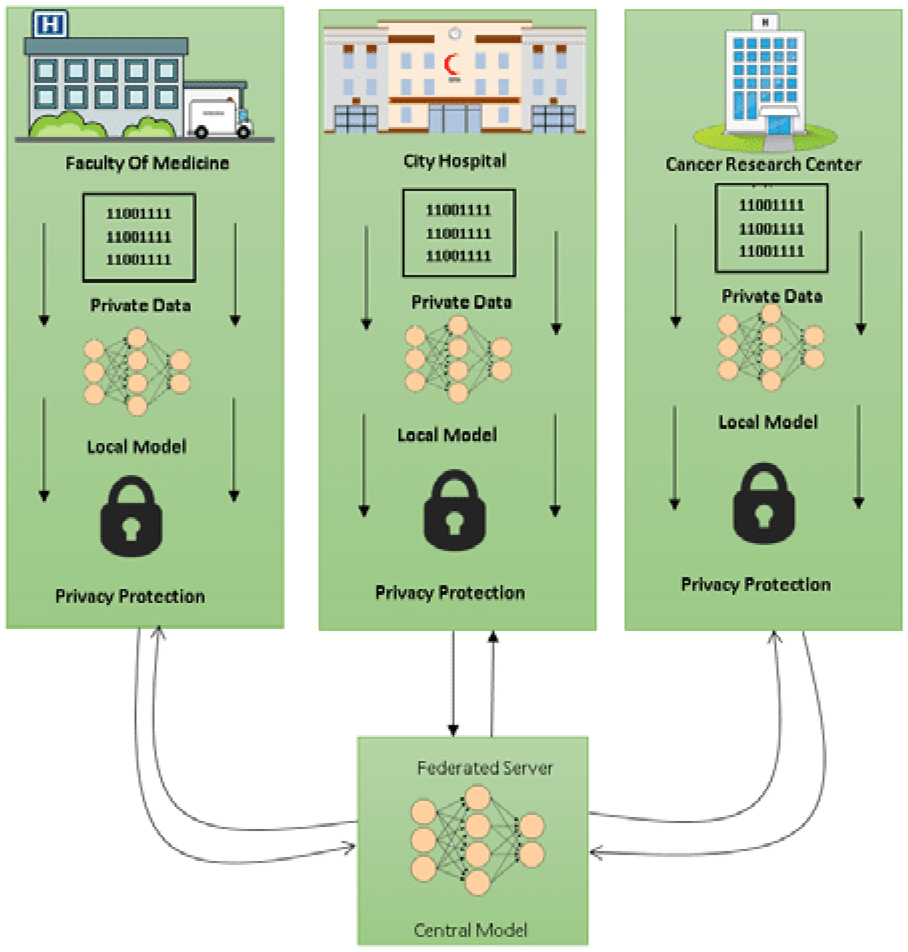
Umami Burger
Phone: (323)263-8626
Address: 738 E 3rd St, Los Angeles, CA 90013

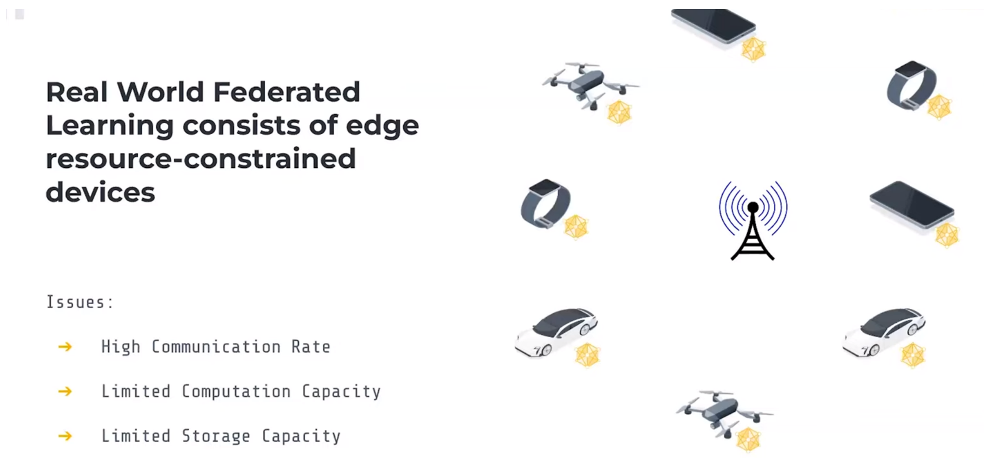
umami burger

Umami Burger
4.3 ★★★★★ 51 Reviews · $
Hamburger Restaurant
Open today: 11AM–11PM

Google Keyboard, locally stores the information and also gives suggestions on next iterations as a query suggestion model.

****





Real World Federated Learning consists of edge resource-constrained devices

Issues:

➔ High Communication Rate

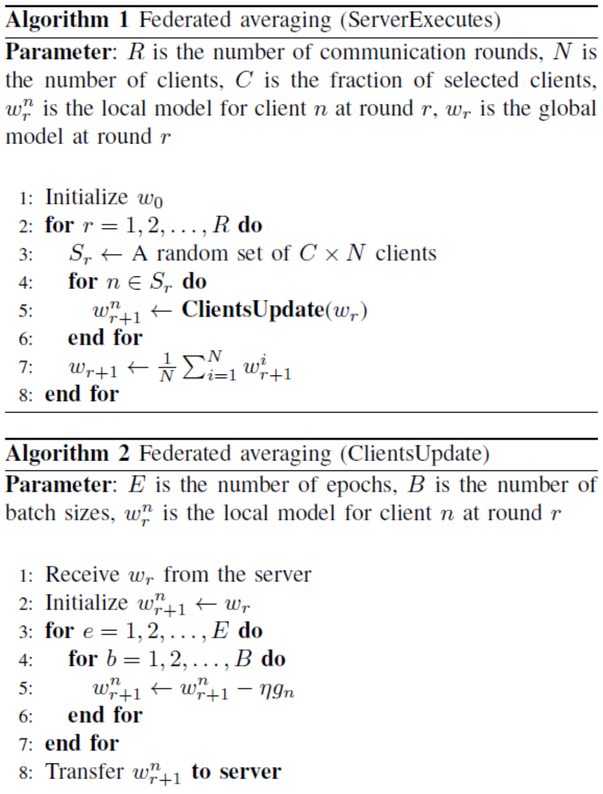
➔ Limited Computation Capacity

➔ Limited Storage Capacity

- In FL the clients transfer only the trained local models instead of the raw data, and it mitigates privacy and network communication issues
- There is still room for improvement in reducing the communication cost of federated learning, since just transferring the trained models still consumes a significant amount of network resources

- Communication costs are frequently a problem in federated learning
- by improving the communication efficiency, the global model on the server can be continuously and efficiently updated, and the performance of federated learning can be improved
- A novel method to reduce the required communication cost for federated learning is by transferring only top updated parameters in neural network models
- We can adjust the criteria of updated parameters to trade off the reduction of communication costs and the loss of model accuracy
- There are two popular approaches to reduce the required communication cost for federated learning: (1) reducing the number of communication rounds, and (2) reducing the amount of communication in each round.
- 1. If the number of communication rounds is reduced, the global model may miss local training information that could have been obtained in the omitted rounds, and thus the model cannot be continuously updated
- 2. Reducing the communication cost in each round can also be divided into two approaches:
    - (1) transferring whole models from selected clients
    - (2) transferring compressed models from all clients
- transferring compressed models is a promising approach because it can balance the trade-off between communication cost and model accuracy by adjusting the compression ratio
- transfer sparse models instead of dense models for reducing the communication cost on both uplink and downlink in federated learning
- **The proposed method in sparse FL Communication constructs a sparse model by selecting only parameters that have been updated significantly**
- **How it works? : We compute the absolute difference between the parameters of the local model before and after training, and exchange only the upper quantile of the updated parameters between the server and the clients.**
- This parameter-wise selection approach increases the opportunity to reduce the communication cost since it omits unnecessary parameters and keeps the necessary parameters for the transfer
- **Justification: It is reasonable not to transfer the parameters that do not have significant updates in the local model, since they may not have much impact on the global model update**.

- the tradeoff between the model accuracy and the communication cost is adjusted with a hyperparameter. This hyperparameter controls the level of sparsification, i.e. what fraction of the model is exchanged between the server and clients
- Terminology:
    - Gradient Descent: Stochastic, Batch, minibatch.
    - Batch gradient descent requires more computation and memory since it needs to process the entire training dataset in each iteration. Stochastic gradient descent, on the other hand, requires less computation and memory since it only needs to process a single example, Mini batch is where we work on a small subset of examples in each iteration.
- Algo: Federated Stochastic Gradient Descent (FedSGD) is a popular conventional federated learning algorithm based on SGD usually used for FL
- The state-of-the-art federated learning algorithm is Federated Averaging (FedAVG).FedAVG was developed to reduce the communication costs of FedSGD.
- The above discussion can be explained with the below algorithm at server and client respectively.



Algorithm 1 Federated averaging (ServerExecutes)

Parameter: R is the number of communication rounds, N is the number of clients, C is the fraction of selected clients, w_r^n is the local model for client n at round r, w_r is the global model at round r

```
1: Initialize $w_0$
2: for $r = 1, 2, \ldots, R$ do
3:   $S_r \leftarrow$ A random set of $C \times N$ clients
4:   for $n \in S_r$ do
5:     $w_{r+1}^n \leftarrow$ ClientsUpdate($w_r$)
6:   end for
7:   $w_{r+1} \leftarrow \frac{1}{N} \sum_{i=1}^{N} w_{r+1}^i$
8: end for
```

Algorithm 2 Federated averaging (ClientsUpdate)

Parameter: E is the number of epochs, B is the number of batch sizes, w_r^n is the local model for client n at round r

```
1: Receive $w_r$ from the server
2: Initialize $w_{r+1}^n \leftarrow w_r$
3: for $e = 1, 2, \ldots, E$ do
4:   for $b = 1, 2, \ldots, B$ do
5:     $w_{r+1}^n \leftarrow w_{r+1}^n - \eta g_n$
6:   end for
7: end for
8: Transfer $w_{r+1}^n$ to server
```

Relation between FedSGD and FedAVG:

FedSGD can be considered as a special case of FedAVG where the local batch size is ∞ and the number of local training epochs is one. In addition, FedAVG allows assigning a fraction of selected clients to reduce the number of participating clients in each communication round.
A few more similar algorithms were proposed, where the algorithms differ in the way of selecting the parameters but the aim of all of them is to reduce the communication cost only.

Future work:
A future work is to investigate other neural network models to improve and reduce the required communication cost for federated learning while maintaining the accuracy of a global model. Updating the parameters in other neural network models should be observed during the local training procedure on each local edge device to reduce communication costs efficiently. In addition, a larger number of edge devices should be used to evaluate the performance of the proposed method.

****

Sparse Autoencoders are one of the most valuable types of Autoencoders. The idea behind Sparse Autoencoders is that we can achieve an information bottleneck (same information with fewer neurons) without reducing the number of neurons in the hidden layers

One immediate benefit of sparse neural networks is smaller size, given we only need to keep the non-zero connections, which is a huge benefit when trying to fit the networks in edge devices. The sparse network is compressed in the same ratio as the amount of sparsity introduced

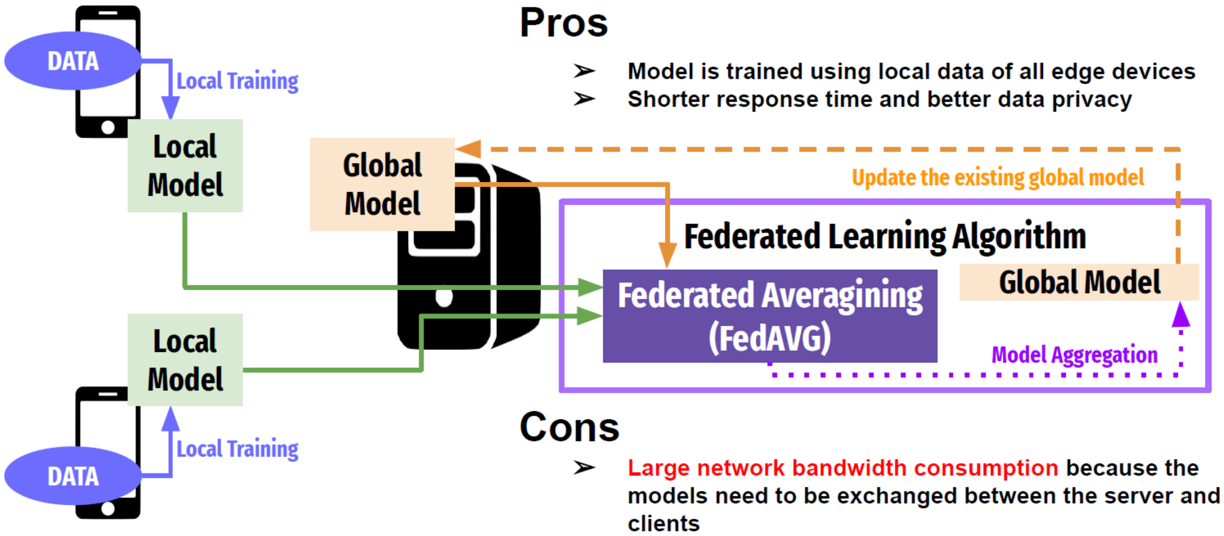
Telemedicine involves the use of telecommunication technologies to provide medical information and services

Watermarking" is the process of hiding digital information in a carrier signal; the hidden information should, but does not need to, contain a relation to the carrier signal.

Watermarks are used to protect images and visual files from being stolen and used or altered without the owner's permission.

****

Federated learning



Pros

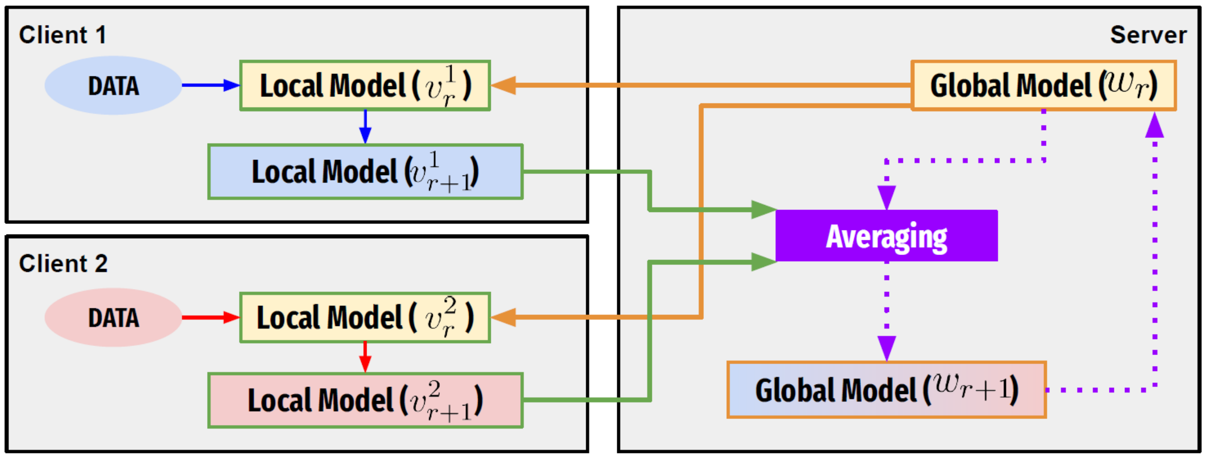
- ➢ Model is trained using local data of all edge devices
- ➢ Shorter response time and better data privacy

Cons

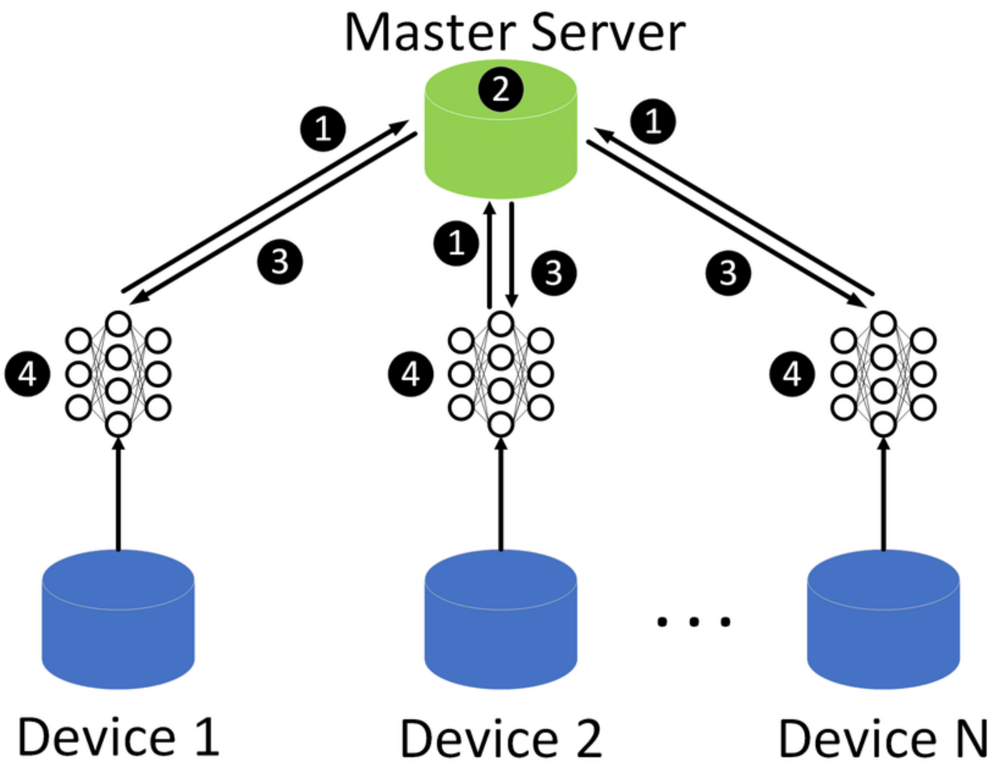
- ➢ Large network bandwidth consumption because the models need to be exchanged between the server and clients

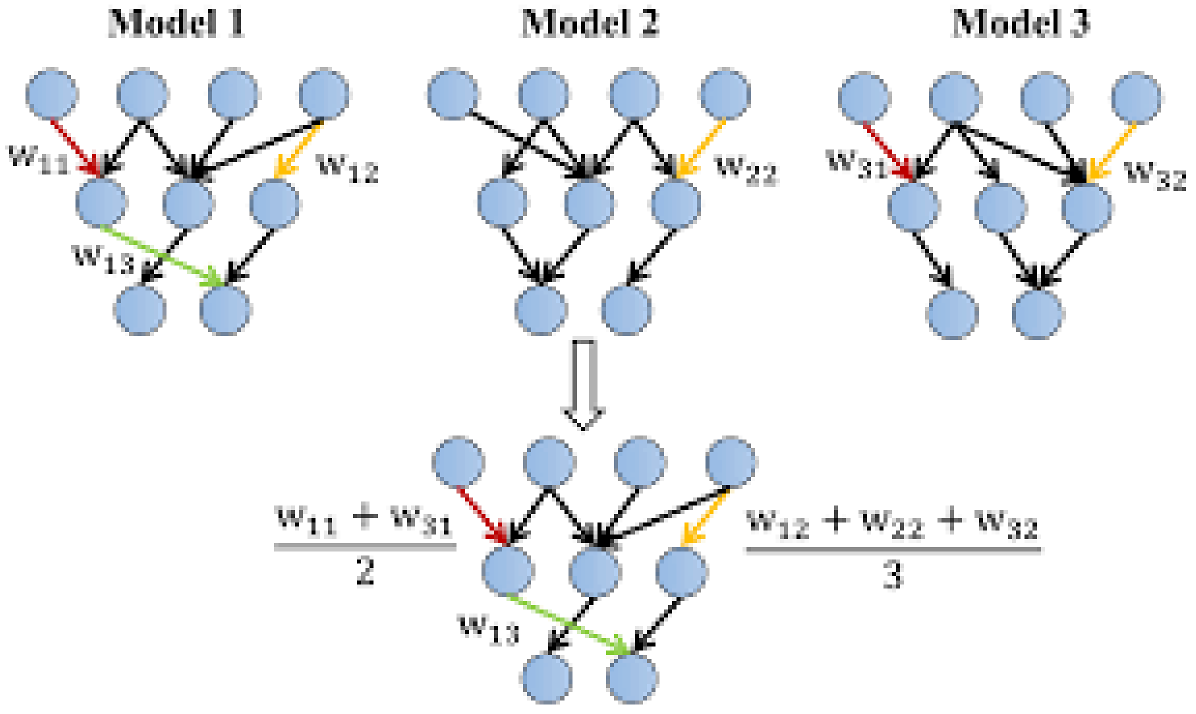
Federated averaging (FedAVG)

w is the weights of the global model on the server
v is the weights of the local model on the client

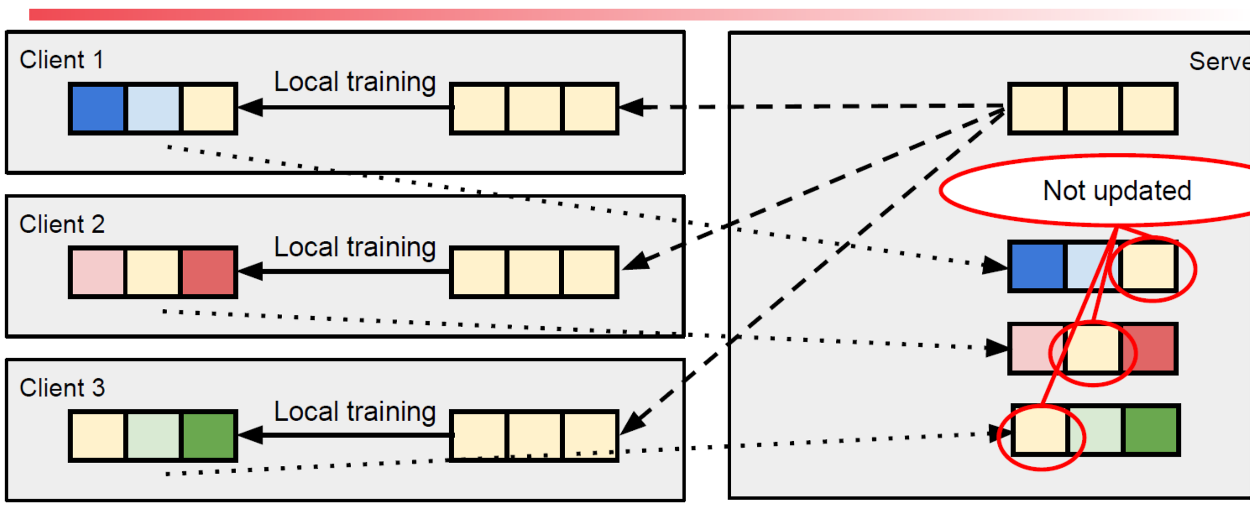


The number of selected clients in each round is $R = C \times N, 0 < C \leq 1,$
where **C** is the fraction of selected clients and **N** is the total number of clients.



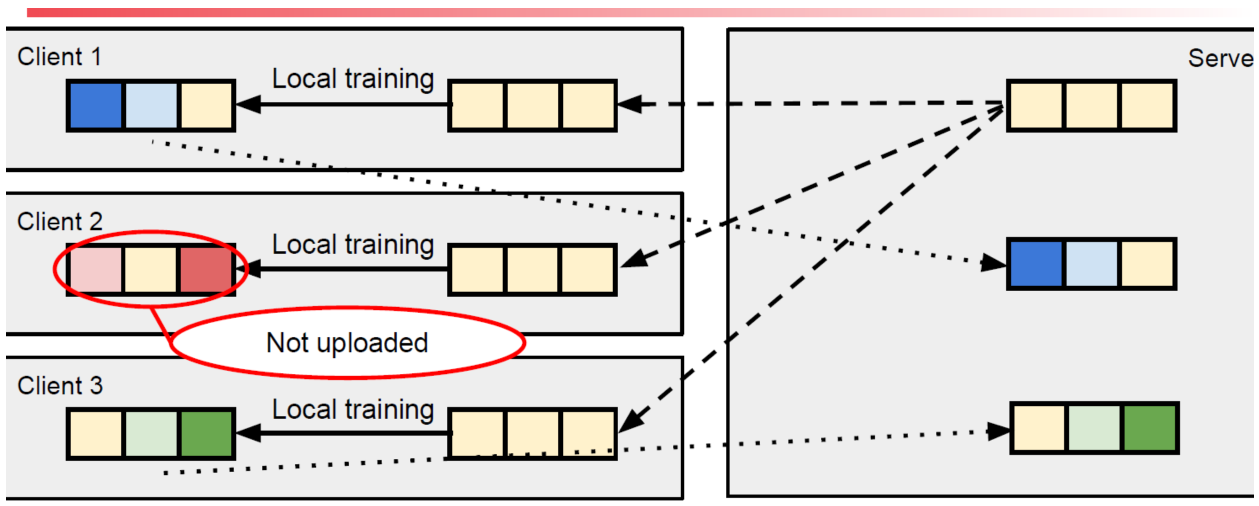


Whole models are exchanged

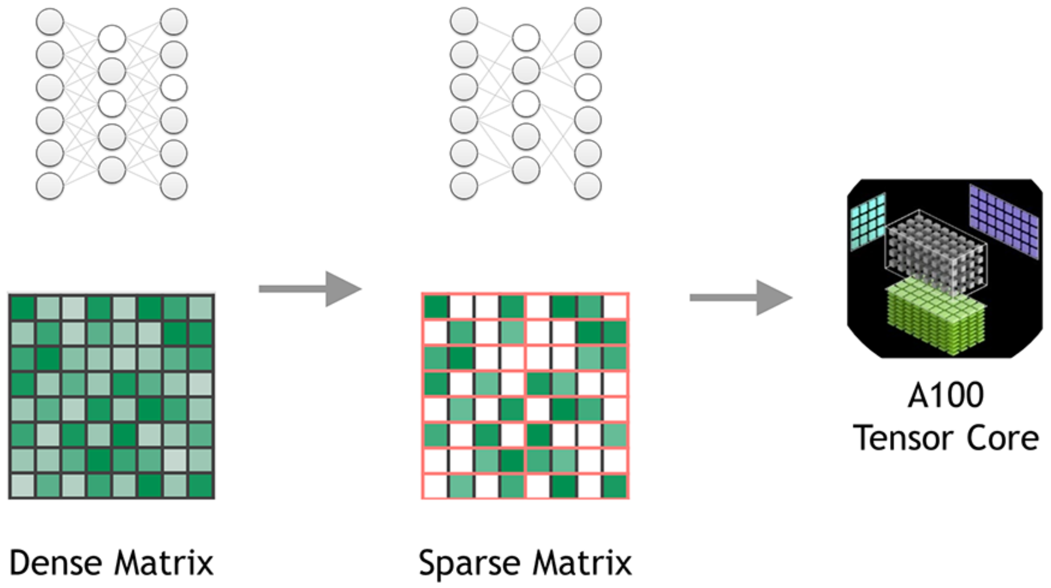


Since the whole models are exchanged between the server and clients, **transferring unupdated parameters** wastes the network bandwidth.

Only a subset of clients participate

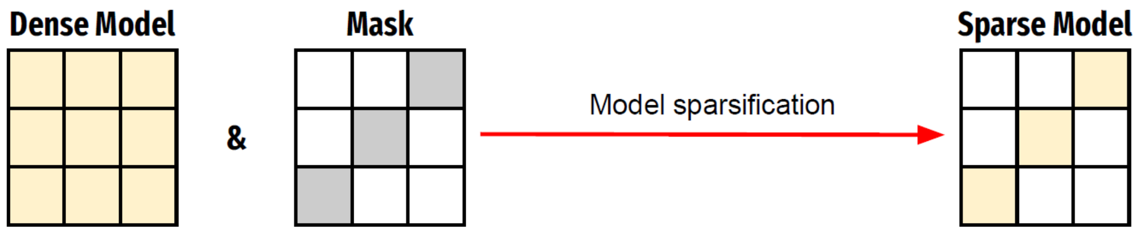


Since only a subset of clients participate in one round, **the server misses local updates** that could have been obtained from the excluded clients.



Model sparsification

Model sparsification **omits some parameters** in the dense model to build a sparse model while keeping the same model architecture

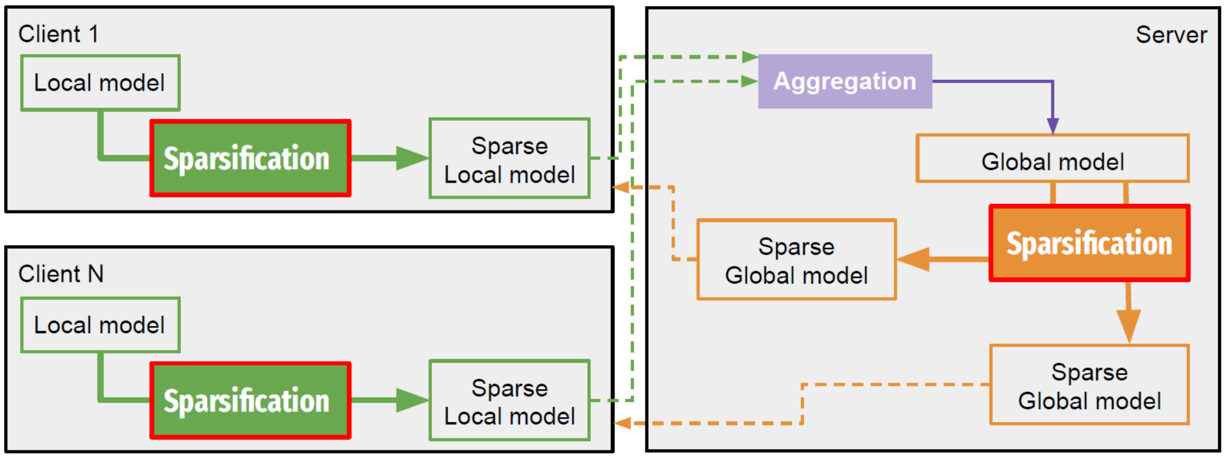


The proposed method exchanges **the most updated parameters** of model between server and clients

Parameters that are significantly changed after training are expected to have **large impact on the model performance**

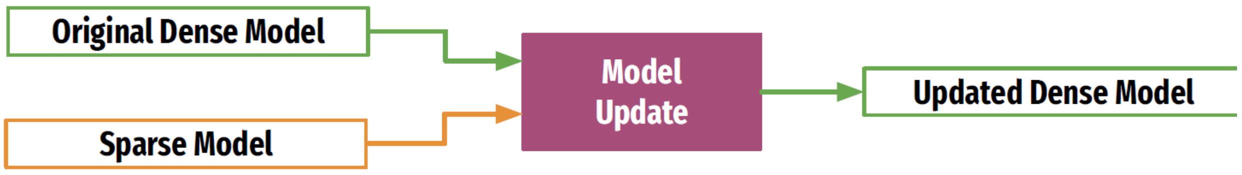
the idea behind the proposed method

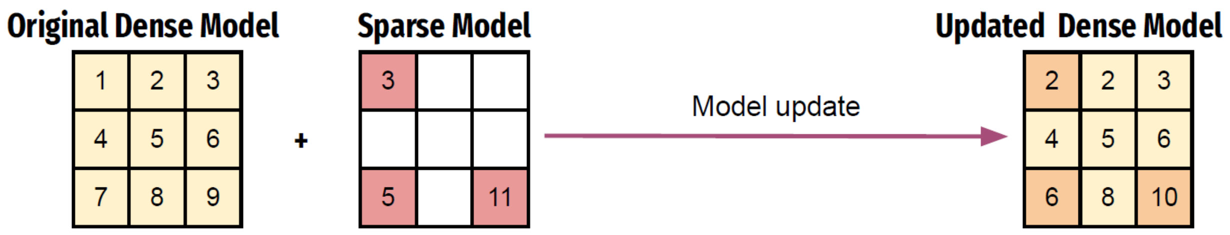
Sparsify the models exchanged between the server and clients in both directions



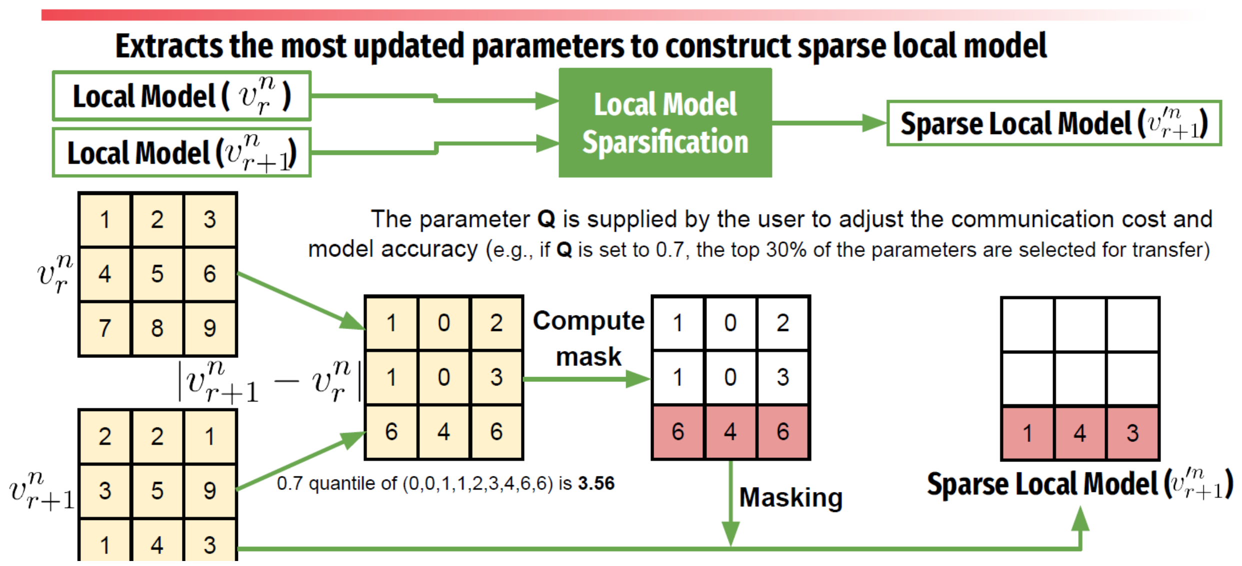
Model update

Updates a dense model using sparse updates sent from the server or clients

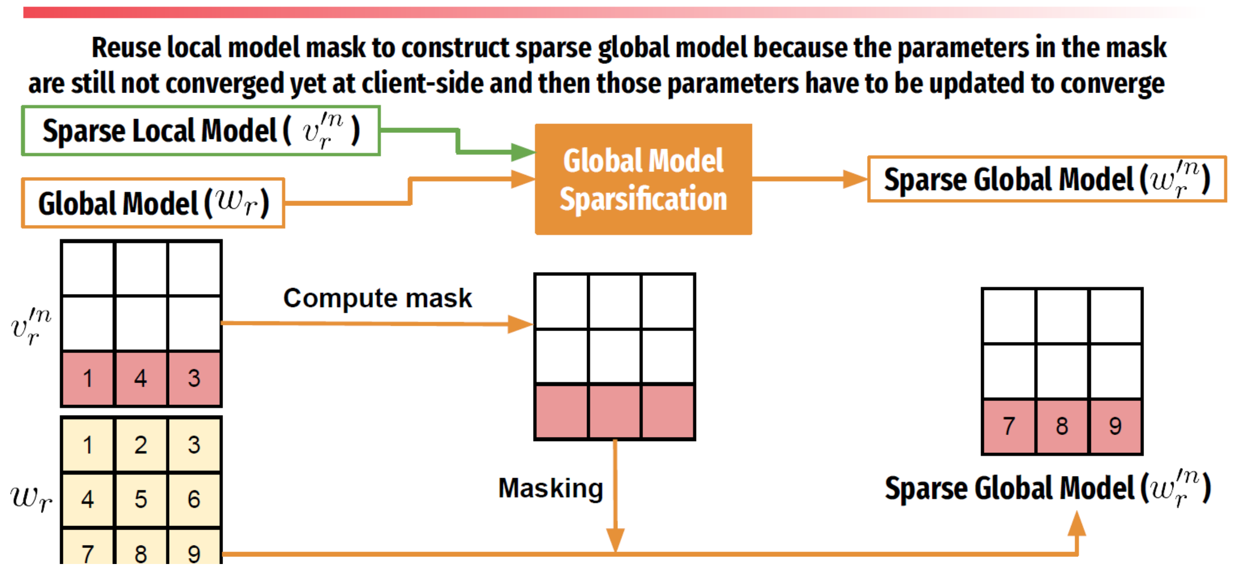




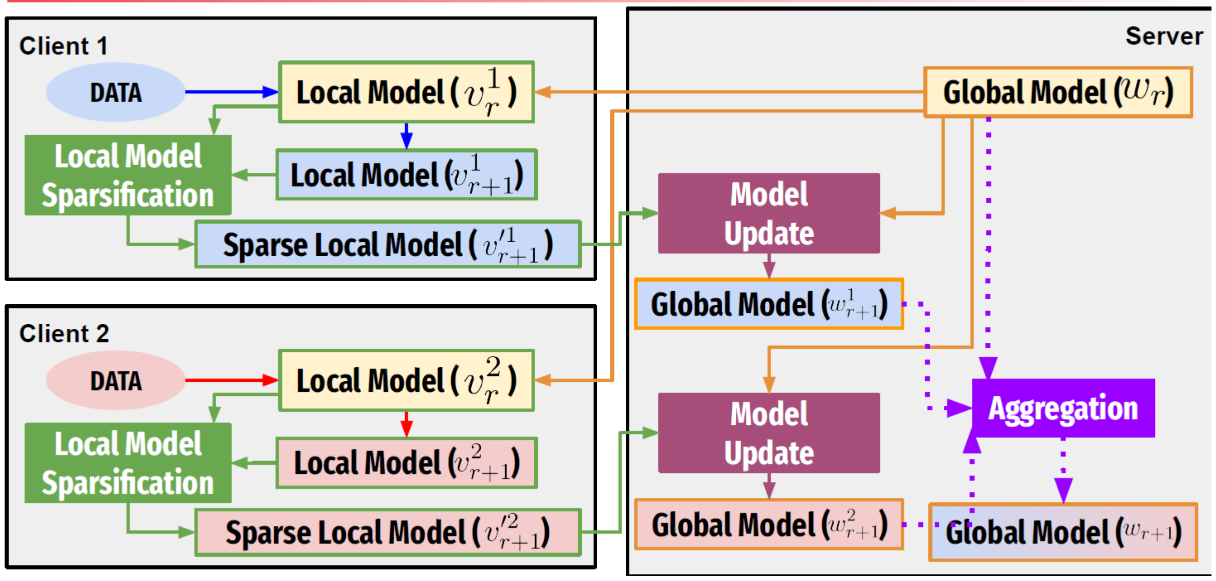
Local model sparsification


Extracts the most updated parameters to construct sparse local model
Local Model (v_r^n)
Local Model (v_{r+1}^n)
Local Model Sparsification
Sparse Local Model ($v_{r+1}'^n$)
The parameter **Q** is supplied by the user to adjust the communication cost and model accuracy (e.g., if **Q** is set to 0.7, the top 30% of the parameters are selected for transfer)
v_r^n
1 2 3 4 5 6 7 8 9
$|v_{r+1}^n - v_r^n|$
1 0 2 1 0 3 6 4 6
Compute mask
1 0 2 1 0 3 6 4 6
v_{r+1}^n
2 2 1 3 5 9 1 4 3
0.7 quantile of (0,0,1,1,2,3,4,6,6) is **3.56**
Masking
1 4 3
Sparse Local Model ($v_{r+1}'^n$)

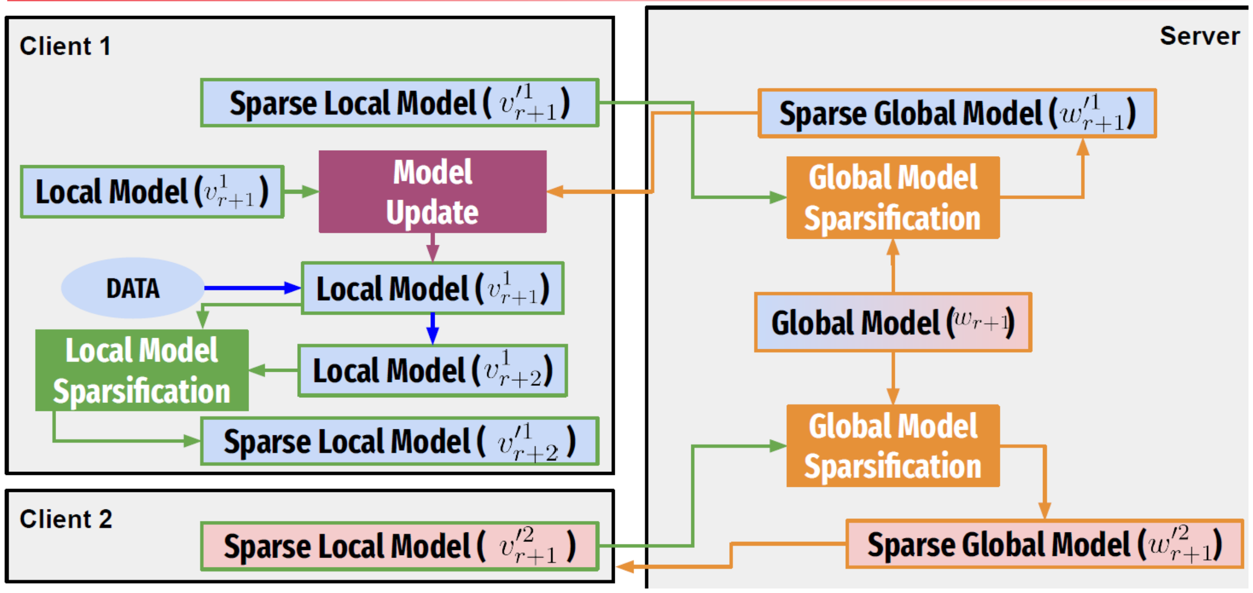
Global model sparsification


Reuse local model mask to construct sparse global model because the parameters in the mask are still not converged yet at client-side and then those parameters have to be updated to converge
Sparse Local Model ($v_r'^n$)
Global Model (w_r)
Global Model Sparsification
Sparse Global Model ($w_r'^n$)
$v_r'^n$
1 4 3
Compute mask
w_r
1 2 3 4 5 6 7 8 9
Masking
7 8 9
Sparse Global Model ($w_r'^n$)

Overview of the proposed method (Uplink)


Client 1
DATA
Local Model (v_r^1)
Local Model Sparsification
Local Model (v_{r+1}^1)
Sparse Local Model (v'^1_{r+1})
Client 2
DATA
Local Model (v_r^2)
Local Model Sparsification
Local Model (v_{r+1}^2)
Sparse Local Model (v'^2_{r+1})
Server
Global Model (w_r)
Model Update
Global Model (w_{r+1}^1)
Model Update
Global Model (w_{r+1}^2)
Aggregation
Global Model (w_{r+1})

Overview of the proposed method (Downlink)


Client 1
Sparse Local Model (v'^1_{r+1})
Local Model (v_{r+1}^1)
Model Update
DATA
Local Model (v_{r+1}^1)
Local Model Sparsification
Local Model (v_{r+2}^1)
Sparse Local Model (v'^1_{r+2})
Client 2
Sparse Local Model (v'^2_{r+1})
Server
Sparse Global Model (w'^1_{r+1})
Global Model Sparsification
Global Model (w_{r+1})
Global Model Sparsification
Sparse Global Model (w'^2_{r+1})

Conclusion

➢ a novel method to reduce the communication cost for federated learning by sparsifying local and global models

➢The proposed method utilises exchanging the most updated parameters

of neural network models